TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN

GIÁO TRÌNH

THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. Làm quen 3

Bài 1) Tạo ứng dụng đầu tiên 3

1.1) Android Studio và Hello World 3

1.2) Giao diện người dùng tương tác đầu tiên 29

1.3) Trình chỉnh sửa bố cục 59

1.4) Văn bản và các chế độ cuộn 101

1.5) Tài nguyên có sẵn 124

Bài 2) Activities 144

2.1) Activity và Intent 144

2.2) Vòng đời của Activity và trạng thái 179

2.3) Intent ngầm định 183

Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ 183

3.1) Trình gỡ lỗi 183

3.2) Kiểm thử đơn vị 183

3.3) Thư viện hỗ trợ 183

CHƯƠNG 2. Trải nghiệm người dùng 184

Bài 1) Tương tác người dùng 184

1.1) Hình ảnh có thể chọn 184

1.2) Các điều khiển nhập liệu 184

1.3) Menu và bộ chọn 184

1.4) Điều hướng người dùng 184

1.5) RecycleView 184

Bài 2) Trải nghiệm người dùng thú vị 184

2.1) Hình vẽ, định kiểu và chủ đề 184

2.2) Thẻ và màu sắc 184

2.3) Bố cục thích ứng 184
Bài 3) Kiểm thử giao diện người dùng 184
3.1) Espresso cho việc kiểm tra UI 184
CHƯƠNG 3. Làm việc trong nền 184
Bài 1) Các tác vụ nền 184
1.1) AsyncTask 184
1.2) AsyncTask và AsyncTaskLoader 184
1.3) Broadcast receivers 184
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền 184
2.1) Thông báo 184
2.2) Trình quản lý cảnh báo 184
2.3) JobScheduler 184
CHƯƠNG 4. Lưu dữ liệu người dùng 185
Bài 1) Tùy chọn và cài đặt 185
1.1) Shared preferences 185
1.2) Cài đặt ứng dụng 185
Bài 2) Lưu trữ dữ liệu với Room 185
2.1) Room, LiveData và ViewModel 185
2.2) Room, LiveData và ViewModel 185

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

Giới thiệu

Trong bài thực hành này, bạn sẽ tìm hiểu cách cài đặt Android Studio, môi trường phát triển Android. Bạn cũng sẽ tạo và chạy ứng dụng Android đầu tiên của mình, Hello World, trên một trình giả lập và trên một thiết bị vật lý.

Những gì Bạn nên biết

Bạn nên có khả năng:

- Hiểu quy trình phát triển phần mềm tổng quát cho các ứng dụng lập trình hướng đối tượng sử dụng một IDE (môi trường phát triển tích hợp) như Android Studio.
- Chứng minh rằng bạn có ít nhất 1-3 năm kinh nghiệm trong lập trình hướng đối tượng, với một phần trong số đó tập trung vào ngôn ngữ lập trình Java. (Các bài thực hành này sẽ không giải thích về lập trình hướng đối tượng hoặc ngôn ngữ Java.

Những gì Bạn sẽ cần:

- Một máy tính chạy Windows hoặc Linux, hoặc một Mac chạy macOS. Xem trang tải xuống Android Studio để biết yêu cầu hệ thống cập nhật.
- Truy cập Internet hoặc một phương pháp thay thế để tải các cài đặt mới nhất của Android Studio và Java lên máy tính của bạn.

Những gì bạn sẽ học

- Cách cài đặt và sử dụng IDE Android Studio.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo một dự án Android từ một mẫu.
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

Những gì bạn sẽ làm

- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.
- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.
- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

Tổng quan về ứng dụng

Sau khi bạn cài thành công Android Studio , bạn sẽ tạo một dự án mới từ một mẫu cho ứng dụng Hello World . Ứng dụng đơn giản này hiển thị chuỗi “Hello World” trên màn hình của máy ảo Android hoặc thiết bị thật.

Ứng dụng khi hoàn thành sẽ trông như thế này:

Bước 1: Cài Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp hoàn chỉnh (IDE), bao gồm một trình chỉnh sửa mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ hỗ trợ phát triển, gỡ lỗi, kiểm thử và tối ưu hiệu suất, giúp quá trình phát triển ứng dụng trở nên nhanh chóng và dễ dàng hơn.

Bạn có thể kiểm thử ứng dụng của mình trên nhiều trình giả lập được cấu hình sẵn hoặc trên thiết bị di động của bạn, đồng thời có thể xây dựng ứng dụng hoàn chỉnh và phát hành trên **Google Play Store**.

Lưu ý: Android Studio liên tục được cải tiến. Để biết thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, hãy truy cập **Android Studio**.

Android Studio khả dụng cho máy tính chạy Windows or Linux , và Macs chạy MacOS. OpenJDK(Java Development Kit) mới nhất được tích hợp sẵn trong Android Studio.

Để bắt đầu với Android Studio , trước tiên kiểm tra yêu cầu hệ thống để đẩm bảm thiết bị của bạn đáp ứng đủ điều kiện . Trình cài đặt tương tự với tất cả nền tảng . Chỉ khác nhau vài điều được ghi dưới đây :

1. Truy cập trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận cấu hình mặc định ở tất cả các bước và đảm bảo rằng tất cả các thành phần cần thiết đều được chọn để cài đặt.
3. Sau khi quá trình cài đặt hoàn tất, Setup Wizard sẽ tự động tải xuống và cài đặt thêm một số thành phần bổ sung, bao gồm Android SDK. Hãy kiên nhẫn, vì quá trình này có thể mất một chút thời gian tùy vào tốc độ Internet của bạn, và một số bước có thể lặp lại.
4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình.

Xử lý sự cố: Nếu gặp vấn đề trong quá trình cài đặt, hãy kiểm tra ghi chú phát hành của Android Studio hoặc nhờ sự trợ giúp từ giảng viên của bạn.

Bước 2 : Tạo ứng dụng Hello World

Trong bước này, bạn sẽ tạo một ứng dụng hiển thị dòng chữ **"Hello World"** để kiểm tra xem Android Studio đã được cài đặt đúng cách chưa, đồng thời làm quen với các bước phát triển cơ bản trong Android Studio.

2.1 Tạo dự án ứng dụng

1. Mở Android Studio nếu chưa mở.
2. Trong cửa sổ chính **Welcome to Android Studio**, nhấn vào **Start a new Android Studio project**.
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** vào ô **Application name**.

4. Kiểm tra **vị trí lưu dự án** mặc định. Nếu cần, thay đổi sang thư mục mà bạn muốn lưu ứng dụng **Hello World** và các dự án khác.

5. Chấp nhận giá trị mặc định **android.example.com** cho **Company Domain**, hoặc nhập một domain riêng nếu muốn.

   Nếu bạn không có ý định xuất bản ứng dụng, có thể giữ nguyên mặc định. Lưu ý rằng nếu bạn thay đổi **package name** của ứng dụng sau này, bạn sẽ phải thực hiện thêm một số bước bổ sung.

6. **Bỏ chọn** các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấn **Next**.

7. Ở màn hình **Target Android Devices**, đảm bảo **Phone and Tablet** được chọn.

   Đặt **API 15: Android 4.0.3 IceCreamSandwich** làm **Minimum SDK**. Nếu API không đúng, hãy dùng menu thả xuống để chọn API phù hợp.

Các bài học trong khóa học này sử dụng các thiết lập này để đảm bảo ứng dụng **Hello World** tương thích với **97% thiết bị Android** trên Google Play Store.

8. Bỏ chọn **Include Instant App support** và tất cả các tùy chọn khác, sau đó nhấn **Next**. Nếu dự án yêu cầu các thành phần bổ sung cho SDK bạn đã chọn, Android Studio sẽ tự động tải xuống và cài đặt chúng.

9. Ở màn hình **Add an Activity**, chọn **Empty Activity**, rồi nhấn **Next**. **Activity** là một màn hình trong ứng dụng Android, thường có một **layout** đi kèm để hiển thị giao diện người dùng. Android Studio cung cấp các mẫu Activity giúp bạn khởi tạo nhanh dự án. Với dự án Hello World , chọn **Empty Activity** như bên dưới và ấn **Next.**

10. Màn hình **Configure Activity** xuất hiện (giao diện này khác nhau tùy thuộc vào mẫu bạn đã chọn ở bước trước). Mặc định, Activity trống được cung cấp bởi mẫu có tên là **MainActivity**. Bạn có thể thay đổi nếu muốn, nhưng bài học này sử dụng **MainActivity**.

11. Đảm bảo **Generate Layout file** được chọn. Tên layout mặc định là **activity_main**, bạn có thể đổi nếu cần, nhưng bài học này sẽ sử dụng activity_main.
12. Đảm bảo **Backwards Compatibility (App Compat)** được chọn để ứng dụng tương thích với các phiên bản Android cũ hơn.
13. Nhấn **Finish**.

Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án với Gradle (quá trình này có thể mất một vài phút).

Mẹo : Xem trang dành cho nhà phát triển **Configure your build** để biết thông tin chi tiết.

Bạn cũng có thể thấy thông báo **"Tip of the day"**, chứa các phím tắt và mẹo hữu ích khác. Nhấn **Close** để đóng thông báo.

Trình chỉnh sửa Android Studio xuất hiện. Thực hiện các bước sau:

1. Nhấp vào tab **activity_main.xml** để mở trình chỉnh sửa giao diện.
2. Nhấp vào tab **Design** trong trình chỉnh sửa giao diện (nếu chưa được chọn) để hiển thị bản xem trước đồ họa của bố cục như hình dưới đây.

3. Nhấp vào tab **MainActivity.java** để xem trình chỉnh sửa mã như hình dưới đây.

2.2 Khám phá Project > Android pane

Trong bài thực hành này, bạn sẽ khám phá cách tổ chức dự án trong Android Studio.

1. Nếu chưa được chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở bên trái cửa sổ Android Studio. Pane **Project** sẽ xuất hiện.
2. Để xem dự án theo hệ thống thư mục tiêu chuẩn của Android, chọn **Android** từ menu thả xuống ở đầu pane **Project**, như hình dưới đây.

Lưu ý: Chương này và các chương khác đề cập đến Project pane, khi được đặt thành **Android**, là **Project > Android** pane.

2.3 Khám phá thư mục Gradle Scripts

Hệ thống build Gradle trong Android Studio giúp dễ dàng thêm các tệp nhị phân bên ngoài hoặc các module thư viện khác vào bản build của bạn dưới dạng dependencies.

Khi bạn tạo một dự án ứng dụng lần đầu tiên, **Project > Android** pane sẽ xuất hiện với thư mục **Gradle Scripts** được mở rộng như hình dưới đây.

Thực hiện các bước sau để khám phá hệ thống Gradle:

1. Nếu thư mục **Gradle Scripts** chưa được mở rộng, hãy nhấp vào biểu tượng tam giác để mở rộng.
   Thư mục này chứa tất cả các tệp cần thiết cho quá trình biên dịch (build) của dự án.

2. Tìm tệp **build.gradle (Project: HelloWorld)**.
   Đây là nơi chứa các tùy chọn cấu hình chung cho tất cả các mô-đun trong dự án của bạn. Mọi dự án trong Android Studio đều có một tệp Gradle build cấp cao nhất. Thông thường, bạn sẽ không cần chỉnh sửa tệp này, nhưng việc hiểu nội dung của nó vẫn rất quan trọng.

   Theo mặc định, tệp build cấp cao nhất sử dụng khối **buildscript** để khai báo kho lưu trữ Gradle và các thư viện phụ thuộc chung cho toàn bộ dự án. Khi dự án cần một thư viện không phải là thư viện cục bộ hoặc tập tin trong máy, Gradle sẽ tìm kiếm thư viện đó trong các kho lưu trữ trực tuyến được xác định trong khối **repositories** của tệp này.

   Mặc định, các dự án mới trong Android Studio sử dụng **JCenter** và **Google** (bao gồm kho Maven của Google) làm kho lưu trữ chính :

3. Tìm tệp **build.gradle(Module: app)**.
   Ngoài tệp build.gradle cấp dự án, mỗi mô-đun đều có một tệp build.gradle riêng, cho phép bạn cấu hình cài đặt biên dịch cho từng mô-đun cụ thể (ứng dụng HelloWorld chỉ có một mô-đun). Việc cấu hình các cài đặt biên dịch này cho phép bạn cung cấp các tùy chọn đóng gói tùy chỉnh, chẳng hạn như các kiểu biên dịch bổ sung và các phiên bản sản phẩm khác nhau. Bạn cũng có thể ghi đè các cài đặt trong tệp AndroidManifest.xml hoặc tệp build.gradle cấp cao nhất.
   Tệp này thường là tệp cần chỉnh sửa khi thay đổi cấu hình cấp ứng dụng, chẳng hạn như khai báo dependencies trong phần dependencies. Bạn có thể khai báo một thư viện phụ thuộc bằng một trong nhiều cấu hình phụ thuộc khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ, câu lệnh: “implementation fileTree(dir: 'libs', include: ['*.jar'])” sẽ thêm tất cả các tệp ".jar" bên trong thư mục libs làm phụ thuộc.
   Dưới đây là tệp **build.gradle(Module: app)** của ứng dụng **HelloWorld**:

4. Nhấn vào biểu tượng tam giác để đóng thư mục **Gradle Scripts**.

2.4 Khám phá thư mục app và res

Tất cả mã nguồn và tài nguyên của ứng dụng nằm trong thư mục **app** và **res**.

1. Mở rộng thư mục **app**, thư mục **java**, và thư mục **com.example.android.helloworld** để xem tệp **MainActivity.java**. Nhấp đúp vào tệp để mở trong trình chỉnh sửa mã.

Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như trong hình trên.

Thư mục **com.example.hello.helloworld** (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp của một gói ứng dụng. Hai thư mục còn lại được sử dụng để kiểm thử và sẽ được đề cập trong bài học khác. Đối với ứng dụng **Hello World**, chỉ có một gói và nó chứa **MainActivity.java**. Tên của Activity đầu tiên mà người dùng nhìn thấy, đồng thời khởi tạo tài nguyên toàn cục của ứng dụng, thường được đặt là **MainActivity** (trong **Project > Android** pane, phần mở rộng tệp bị ẩn).

2. Mở rộng thư mục **res**, sau đó mở thư mục **layout**, và nhấp đúp vào tệp **activity_main.xml** để mở nó trong trình chỉnh sửa giao diện.

Thư mục **res** chứa các tài nguyên như bố cục (layouts), chuỗi ký tự (strings) và hình ảnh (images). Một **Activity** thường đi kèm với một bố cục giao diện người dùng (UI views) được định nghĩa dưới dạng tệp XML. Tệp này thường được đặt tên theo **Activity** tương ứng.

2.5 Khám phá thư mục manifests

Thư mục manifests chứa các tệp cung cấp thông tin cần thiết về ứng dụng của bạn cho hệ thống Android, thông tin này phải có trước khi hệ thống có thể chạy bất kỳ mã nào của ứng dụng.

1. Mở rộng thư mục manifests.
2. Mở tệp AndroidManifest.xml.

Tệp AndroidManifest.xml mô tả tất cả các thành phần của ứng dụng Android. Mọi thành phần của ứng dụng, chẳng hạn như từng Activity, phải được khai báo trong tệp XML này. Trong các bài học khác của khóa học, bạn sẽ chỉnh sửa tệp này để thêm tính năng và quyền truy cập. Để tìm hiểu tổng quan, hãy xem App Manifest Overview.

Bước 3 : Sử dụng trình giả lập

Trong nhiệm vụ này, bạn sẽ sử dụng trình quản lý Android Virtual Device (AVD) để tạo một thiết bị ảo (còn gọi là trình giả lập) mô phỏng cấu hình của một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng Android Emulator có các yêu cầu bổ sung ngoài các yêu cầu hệ thống cơ bản cho Android Studio.

Bằng cách sử dụng AVD Manager, bạn có thể xác định các đặc điểm phần cứng của thiết bị, mức API, bộ nhớ, giao diện và các thuộc tính khác, sau đó lưu chúng dưới dạng một thiết bị ảo. Với các thiết bị ảo, bạn có thể kiểm thử ứng dụng trên nhiều cấu hình thiết bị khác nhau (chẳng hạn như máy tính bảng và điện thoại) với các mức API khác nhau mà không cần sử dụng thiết bị vật lý.

3.1 Tạo một thiết bị ảo Android (AVD)

Để chạy trình giả lập trên máy tính, bạn cần tạo một cấu hình mô tả thiết bị ảo.

1. Trong Android Studio, chọn **Tools > Android > AVD Manager**, hoặc nhấp vào biểu tượng **AVD Manager** trên thanh công cụ. Màn hình **Your Virtual Devices** xuất hiện. Nếu bạn đã tạo các thiết bị ảo trước đó, chúng sẽ được hiển thị trên màn hình (như trong hình minh họa); nếu chưa, danh sách sẽ trống.

2. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng cung cấp các cột thông tin bao gồm: **Kích thước đường chéo màn hình** (*Size*), **Độ phân giải màn hình (pixel)** (*Resolution*), và **Mật độ điểm ảnh** (*Density*).

3. Chọn một thiết bị như **Nexus 5X** hoặc **Pixel XL**, và nhấp vào **Next**. Màn hình **System Image** xuất hiện.
4. Nhấp vào tab **Recommended** nếu nó chưa được chọn, và chọn phiên bản hệ điều hành Android để chạy trên thiết bị ảo (ví dụ: **Oreo**).

Có nhiều phiên bản khác ngoài những phiên bản hiển thị trong tab **Recommended**. Hãy xem các tab **x86 Images** và **Other Images** để tìm thêm. Nếu có liên kết **Download** bên cạnh một system image bạn muốn sử dụng, điều đó có nghĩa là nó chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống và nhấp vào **Finish** khi hoàn tất.

5. Sau khi chọn system image, nhấp vào **Next**. Màn hình **Android Virtual Device (AVD)** xuất hiện. Bạn cũng có thể thay đổi tên của AVD. Kiểm tra cấu hình và nhấp vào **Finish**.

3.2 Chạy ứng dụng trên thiết bị ảo

Trong nhiệm vụ này , bạn sẽ hoàn thành chạy ứng dụng Hello World của bạn.

1. Trong Android Studio, chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run** trên thanh công cụ.
2. Trong cửa sổ **Select Deployment Target**, dưới **Available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo và nhấp vào **OK**.

Trình giả lập khởi động và chạy giống như một thiết bị vật lý. Tùy thuộc vào tốc độ của máy tính, quá trình này có thể mất một khoảng thời gian. Ứng dụng của bạn sẽ được biên dịch, và khi trình giả lập sẵn sàng, Android Studio sẽ tải ứng dụng lên trình giả lập và chạy nó.

Bạn sẽ thấy ứng dụng **Hello World** hiển thị như trong hình minh họa sau.

Mẹo: Khi kiểm thử trên thiết bị ảo, bạn nên khởi động nó một lần ngay từ đầu phiên làm việc. Không nên đóng trình giả lập cho đến khi hoàn tất kiểm thử ứng dụng để tránh phải khởi động lại thiết bị. Để đóng thiết bị ảo, nhấn nút **X** ở góc trên của trình giả lập, chọn **Quit** từ menu hoặc nhấn **Control-Q** trên Windows hoặc **Command-Q** trên macOS.

Bước 4 : (Tùy chọn) Sử dụng thiết bị thật

Trong nhiệm vụ cuối cùng này, bạn sẽ chạy ứng dụng trên một thiết bị di động vật lý như điện thoại hoặc máy tính bảng. Bạn nên luôn kiểm thử ứng dụng trên cả thiết bị ảo và thiết bị vật lý.

Những gì bạn cần:

- Một thiết bị Android như điện thoại hoặc máy tính bảng.
- Cáp dữ liệu để kết nối thiết bị Android với máy tính qua cổng USB.
- Nếu bạn sử dụng hệ điều hành Linux hoặc Windows, có thể cần thực hiện thêm một số bước để chạy ứng dụng trên thiết bị phần cứng. Hãy kiểm tra tài liệu Using Hardware Devices. Bạn cũng có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Đối với trình điều khiển USB trên Windows, hãy xem OEM USB Drivers.

4.1 Bật USB debugging

Để Android Studio có thể giao tiếp với thiết bị của bạn, bạn cần bật **USB Debugging** trên thiết bị Android. Tùy chọn này được kích hoạt trong **Developer options** của thiết bị.

Trên Android 4.2 trở lên, màn hình **Developer options** bị ẩn theo mặc định. Để hiển thị và bật **USB Debugging**, hãy làm theo các bước sau:

1. Trên thiết bị, mở **Cài đặt (Settings)**, tìm **Giới thiệu về điện thoại (About phone)**, nhấn vào đó và chạm vào **Số bản dựng (Build number)** bảy lần.
2. Quay lại màn hình trước (**Cài đặt / Hệ thống** - *Settings / System*). Tùy chọn **Developer options** sẽ xuất hiện trong danh sách. Nhấn vào **Developer options**.
3. Bật **USB Debugging**.

4.2 Chạy ứng dụng của bạn trên thiết bị

Bây giờ bạn có thể kết nối thiết bị và chạy ứng dụng từ Android Studio.

1. Kết nối thiết bị với máy tính bằng cáp USB.
2. Nhấn nút **Run** trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ mở ra, hiển thị danh sách các trình giả lập và thiết bị được kết nối.
3. Chọn thiết bị của bạn và nhấn **OK**.

Android Studio sẽ cài đặt và chạy ứng dụng trên thiết bị của bạn.

Khắc phục sự cố

Nếu Android Studio không nhận diện được thiết bị của bạn, hãy thử các bước sau:

1. Rút và cắm lại thiết bị.
2. Khởi động lại Android Studio.

Nếu máy tính vẫn không tìm thấy thiết bị hoặc báo lỗi "unauthorized", hãy làm theo các bước sau:

1. Rút kết nối thiết bị.
2. Trên thiết bị, mở **Developer Options** trong ứng dụng **Settings**.
3. Nhấn **Revoke USB Debugging authorizations**.

4. Kết nối lại thiết bị với máy tính.
5. Khi được nhắc, cấp quyền xác thực.

Bạn có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Xem tài liệu **Using Hardware Devices** để biết thêm chi tiết.

Bước 5: Thay đổi cấu hình Gradle của ứng dụng

Trong nhiệm vụ này, bạn sẽ thay đổi một số cấu hình của ứng dụng trong tệp **build.gradle(Module:app)** để tìm hiểu cách thực hiện thay đổi và đồng bộ chúng với dự án Android Studio của bạn.

5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện theo các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa được mở và nhấp đúp vào tệp **build.gradle(Module:app)**. Nội dung của tệp sẽ xuất hiện trong trình chỉnh sửa mã.
2. Trong khối **defaultConfig**, thay đổi giá trị của **minSdkVersion** thành **17** như bên dưới (giá trị ban đầu là 15).

Trình chỉnh sửa mã hiển thị một thanh thông báo ở trên cùng với liên kết **Sync Now**.

5.2 Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi trong các tệp cấu hình build của dự án, Android Studio yêu cầu bạn đồng bộ hóa các tệp dự án để có thể nhập các thay đổi cấu hình build và chạy một số kiểm tra nhằm đảm bảo cấu hình không gây ra lỗi build.

Để đồng bộ các tệp dự án, nhấp vào **Sync Now** trong thanh thông báo xuất hiện khi bạn thực hiện thay đổi (như trong hình trước đó) hoặc nhấp vào biểu tượng **Sync Project with Gradle Files** trên thanh công cụ.

Khi quá trình đồng bộ Gradle hoàn tất, thông báo **Gradle build finished** sẽ xuất hiện ở góc dưới bên trái cửa sổ Android Studio.

Để tìm hiểu sâu hơn về Gradle, hãy tham khảo tài liệu **Build System Overview** và **Configuring Gradle Builds**.

Bước 6: Thêm các câu lệnh log vào ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh Log vào ứng dụng của mình, hiển thị thông báo trong bảng Logcat. Thông báo Log là một công cụ gỡ lỗi mạnh mẽ mà bạn có thể sử dụng để kiểm tra giá trị, luồng thực thi và báo cáo ngoại lệ.

6.1 Xem bảng Logcat

Để xem bảng Logcat, hãy nhấp vào tab Logcat ở phía dưới cửa sổ Android Studio, như minh họa trong hình dưới đây.

Trong hình trên:

1. Tab **Logcat** để mở và đóng bảng Logcat, hiển thị thông tin về ứng dụng khi đang chạy. Nếu bạn thêm các câu lệnh Log vào ứng dụng, thông báo Log sẽ xuất hiện ở đây.
2. Menu mức Log được đặt thành **Verbose** (mặc định), hiển thị tất cả thông báo Log. Các cài đặt khác bao gồm **Debug, Error, Info** và **Warn**.

6.2 Thêm câu lệnh Log vào ứng dụng của bạn

Các câu lệnh Log trong mã ứng dụng của bạn hiển thị thông báo trong bảng Logcat. Ví dụ: **Log.d("MainActivity", "Hello World");**

Các phần của thông báo gồm:

- **Log**: Lớp Log dùng để gửi thông báo nhật ký đến bảng Logcat.
- **d**: Mức nhật ký Debug để lọc hiển thị thông báo trong bảng Logcat. Các mức nhật ký khác bao gồm **e** cho Error, **w** cho Warn và **i** cho Info.
- **"MainActivity"**: Đối số đầu tiên là một thẻ (tag) có thể được sử dụng để lọc thông báo trong bảng Logcat. Thông thường, đây là tên của **Activity** nơi thông báo được tạo ra. Tuy nhiên, bạn có thể đặt bất kỳ giá trị nào hữu ích cho quá trình gỡ lỗi.
  Theo quy ước, các thẻ log được định nghĩa dưới dạng hằng số trong **Activity**:
  private static final String LOG_TAG = MainActivity.class.getSimpleName();

- **"Hello world"**: Đối số thứ hai là thông báo thực tế.

Làm theo các bước sau:

1. Mở ứng dụng Hello World trong Android Studio và mở **MainActivity**.
2. Để tự động thêm các import không mơ hồ vào dự án của bạn (chẳng hạn như android.util.Log cần thiết để sử dụng Log), chọn **File > Settings** trong Windows hoặc **Android Studio > Preferences** trong macOS.
3. Chọn **Editor > General > Auto Import**. Chọn tất cả các hộp kiểm và đặt **Insert imports on paste** thành **All**.

4. Nhấp vào **Apply**, sau đó nhấp vào **OK**.
5. Trong phương thức onCreate() của **MainActivity**, thêm câu lệnh sau:

   Log.d("MainActivity", "Hello World");

   Phương thức onCreate() bây giờ sẽ trông như đoạn mã sau:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);
    Log.d( tag: "MainActivity",  msg: "Hello World");
```

6. Nếu Logcat chưa mở, hãy nhấp vào tab Logcat ở cuối cửa sổ Android Studio để mở nó.
7. Kiểm tra xem tên mục tiêu và tên gói của ứng dụng có đúng không.
8. Thay đổi mức Log trong Logcat thành Debug (hoặc giữ nguyên là Verbose vì có rất ít thông báo log).
9. Chạy ứng dụng của bạn.

Thông báo sau đây sẽ xuất hiện trong Logcat:

```
2025-02-21 21:53:57.899  8125-8125  MainActivity          com.example.helloworld          D  Hello World
```

Tổng kết

- Để cài đặt Android Studio, truy cập Android Studio và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo ứng dụng mới, đảm bảo rằng API 15: Android 4.0.3 IceCreamSandwich được đặt làm Minimum SDK.
- Để xem cấu trúc phân cấp Android của ứng dụng trong ngăn Project, nhấp vào tab Project trong cột tab dọc, sau đó chọn Android trong menu bật lên ở trên cùng.
- Chỉnh sửa tệp build.gradle(Module:app) khi cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.

Bài tập về nhà

Xây dựng và chạy một ứng dụng

- Tạo một dự án Android mới từ mẫu **Empty Template**.
- Thêm các câu lệnh ghi log với nhiều mức khác nhau trong phương thức onCreate() của **MainActivity**.
- Tạo một trình giả lập (emulator) cho một thiết bị, nhắm mục tiêu bất kỳ phiên bản Android nào bạn muốn, và chạy ứng dụng.
- Sử dụng bộ lọc trong **Logcat** để tìm các câu lệnh ghi log của bạn và điều chỉnh mức lọc để chỉ hiển thị các câu lệnh ghi log ở mức **debug** hoặc **error**.

Trả lời câu hỏi

Câu hỏi 1

Tên của tệp layout cho activity chính là gì?

- MainActivity.java
- AndroidManifest.xml
- **activity_main.xml**
- build.gradle

Câu hỏi 2

Tên của resource chuỗi nào xác định tên của ứng dụng?

- **app_name**
- xmlns:app
- android:name
- applicationId

Câu hỏi 3

Bạn sử dụng công cụ nào để tạo một trình giả lập (emulator) mới?

- Android Device Monitor
- **AVD Manager**
- SDK Manager
- Theme Editor

Câu hỏi 4

Giả sử ứng dụng của bạn bao gồm câu lệnh ghi log sau:

Log.i("MainActivity", "MainActivity layout is complete");

Bạn sẽ thấy câu lệnh "MainActivity layout is complete" trong bảng Logcat nếu menu mức log được đặt thành mức nào sau đây? (Gợi ý: Có thể có nhiều đáp án đúng.)

- **Verbose**
- Debug
- **Info**
- Warn
- Error
- Assert

Nộp ứng dụng của bạn để chấm điểm

Hãy kiểm tra để đảm bảo ứng dụng có các yêu cầu sau:

- Một Activity hiển thị dòng chữ "Hello World" trên màn hình.
- Các câu lệnh ghi log trong phương thức onCreate() của activity chính.
- Mức log trong bảng Logcat chỉ hiển thị các câu lệnh ghi log ở mức debug hoặc error.

1.2) Giao diện người dùng tương tác đầu tiên

Mở đầu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là views — mọi phần tử trên màn hình đều là một View. Lớp View đại diện cho khối xây dựng cơ bản của tất cả các thành phần giao diện người dùng và là lớp cơ sở cho các lớp cung cấp các thành phần giao diện người dùng tương tác như nút bấm, hộp kiểm và trường nhập văn bản. Các lớp View con thường được sử dụng được mô tả trong nhiều bài học bao gồm:

- TextView để hiển thị văn bản.
- EditText cho phép người dùng nhập và chỉnh sửa văn bản.
- Button và các phần tử có thể nhấp khác (chẳng hạn như RadioButton, CheckBox và Spinner) để cung cấp hành vi tương tác.
- ScrollView và RecyclerView để hiển thị các mục có thể cuộn.
- ImageView để hiển thị hình ảnh.
- ConstraintLayout và LinearLayout để chứa các phần tử View khác và định vị chúng.

Mã Java hiển thị và điều khiển giao diện người dùng (UI) được chứa trong một lớp mở rộng từ Activity. Một Activity thường được liên kết với một bố cục giao diện người dùng được định nghĩa dưới dạng tệp XML (eXtended Markup Language). Tệp XML này thường được đặt tên theo Activity của nó và xác định bố cục của các phần tử View trên màn hình.

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bố cục được định nghĩa trong tệp bố cục activity_main.xml, trong đó có một TextView với nội dung "Hello World".

Trong các ứng dụng phức tạp hơn, một Activity có thể thực hiện các hành động để phản hồi thao tác chạm của người dùng, vẽ nội dung đồ họa hoặc yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet. Bạn sẽ tìm hiểu thêm về lớp Activity trong một bài học khác.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên — một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo một ứng dụng sử dụng mẫu "Empty Activity". Bạn cũng sẽ học cách sử dụng trình chỉnh sửa bố cục để thiết kế giao diện

và chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng này để hoàn thành các bài thực hành khác trong khóa học này.

Những điều Bạn nên biết

Bạn nên làm quen với:

- Cách cài đặt và mở Android Stuido
- Cách tạo ứng dụng HelloWorld
- Cách chạy ứng dụng HelloWorld\

Những gì Bạn gì học

- Cách tạo một ứng dụng có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục để thiết kế giao diện.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới. Hãy tham khảo bảng thuật ngữ và khái niệm để có các định nghĩa dễ hiểu.

Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử Button cùng một TextView vào bố cục.
- Điều chỉnh từng phần tử trong ConstraintLayout để ràng buộc chúng với lề và các phần tử khác.
- Thay đổi thuộc tính của các phần tử giao diện người dùng (UI).
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi được mã hóa cứng thành tài nguyên chuỗi.\
- Triển khai phương thức xử lý sự kiện nhấp để hiển thị thông báo trên màn hình khi người dùng nhấn vào từng Button.

Tổng quan ứng dụng

Ứng dụng HelloToast bao gồm hai phần tử Button và một TextView. Khi người dùng nhấn vào Button đầu tiên, ứng dụng sẽ hiển thị một thông báo ngắn (Toast) trên màn hình. Nhấn vào Button thứ hai sẽ tăng bộ đếm số lần nhấp hiển thị trong TextView, bắt đầu từ số không.

Dưới đây là giao diện hoàn chỉnh của ứng dụng:

08:16
100%
Hello Toast
Toast
10
Count

Bước 1 : Tạo và khám phá một dự án mới

Trong bài thực hành này, bạn sẽ thiết kế và triển khai một dự án cho ứng dụng HelloToast. Một liên kết đến mã giải pháp sẽ được cung cấp ở cuối.

1.1 Tạo dự án Android Studio

Khởi động Android Studio và tạo một dự án mới với các thông số sau:

Application Name	**Hello Toast**
Company Name	**com.example.android** (or your own domain)
Phone and Tablet Minimum SDK	**API15: Android 4.0.3 IceCreamSandwich**
Template	**Empty Activity**
Generate Layout file box	Selected
Backwards Compatibility box	Selected

Chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run** trên thanh công cụ để biên dịch và chạy ứng dụng trên trình giả lập hoặc thiết bị của bạn.

1.2 Khám phá trình chỉnh sửa bố cục

Android Studio cung cấp trình chỉnh sửa bố cục (layout editor) để nhanh chóng xây dựng giao diện người dùng (UI) của ứng dụng. Nó cho phép bạn kéo các phần tử vào chế độ thiết kế trực quan và chế độ xem bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc (constraints) và đặt thuộc tính.Ràng buộc xác định vị trí của một phần tử UI trong bố cục. Một ràng buộc đại diện cho một kết nối hoặc căn chỉnh với một view khác, bố cục cha hoặc một hướng dẫn vô hình.

Khám phá trình chỉnh sửa bố cục và tham khảo hình minh họa bên dưới khi thực hiện theo các bước được đánh số.

1. Trong ngăn **Project > Android**, điều hướng đến thư mục **app > res > layout**, sau đó nhấp đúp vào tệp **activity_main.xml** để mở, nếu nó chưa được mở.

2. Nhấp vào tab **Design** nếu nó chưa được chọn. Bạn sử dụng tab **Design** để thao tác các phần tử và bố cục, và tab **Text** để chỉnh sửa mã XML của bố cục.

3. Ngăn **Palette** hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng.

4. Ngăn **Component Tree** hiển thị cây phân cấp của các phần tử UI. Các phần tử View được tổ chức theo cấu trúc cây gồm cha và con, trong đó phần tử con kế thừa thuộc tính của phần tử cha. Trong hình minh họa, **TextView** là phần tử con của **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về các phần tử này sau trong bài học.

5. Các ngăn **Design** và **Blueprint** trong trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ chứa một phần tử: một **TextView** hiển thị dòng chữ "Hello World".

6. Tab **Attributes** hiển thị bảng **Attributes**, nơi bạn có thể đặt các thuộc tính cho một phần tử UI.

Mẹo: Xem **Building a UI with Layout Editor** để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục, và **Meet Android Studio** để xem toàn bộ tài liệu về Android Studio.

Bước 2 : Thêm các phần tử View trong trình chỉnh sửa bố cục

Trong nhiệm vụ này, bạn sẽ tạo bố cục UI cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng cách sử dụng các tính năng của ConstraintLayout.Bạn có thể tạo các ràng buộc (constraints) theo cách thủ công, như sẽ được trình bày sau, hoặc tự động bằng công cụ Autoconnect.

2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện các bước sau:

1. Mở **activity_main.xml** từ **Project > Android** pane nếu nó chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào nó.

   Nếu không có bản thiết kế (**blueprint**), nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**.

2. Công cụ **Autoconnect** cũng nằm trên thanh công cụ. Nó được bật theo mặc định. Đối với bước này, hãy đảm bảo công cụ không bị vô hiệu hóa.

3. Nhấp vào nút **Zoom in** để phóng to bảng thiết kế và bản thiết kế để xem cận cảnh.

4. Chọn **TextView** trong bảng **Component Tree**. **TextView** hiển thị "Hello World" sẽ được tô sáng trong bảng thiết kế và bản thiết kế, đồng thời các ràng buộc của phần tử sẽ hiển thị.

5. Thực hiện theo hình minh họa bên dưới cho bước này. Nhấp vào tay cầm tròn ở bên phải của **TextView** để xóa ràng buộc ngang ràng buộc phần tử này với cạnh phải của bố cục. **TextView** sẽ di chuyển sang bên trái vì nó không còn bị ràng buộc với cạnh phải nữa. Để thêm lại ràng buộc ngang, nhấp vào cùng tay cầm đó và kéo một đường đến cạnh phải của bố cục.

Trong bảng thiết kế hoặc bảng bản vẽ, các tay cầm sau xuất hiện trên phần tử TextView:

- Tay cầm ràng buộc: Để tạo một ràng buộc như trong hình minh họa động ở trên, nhấp vào tay cầm ràng buộc, được hiển thị dưới dạng một vòng tròn ở cạnh của phần tử. Sau đó, kéo tay cầm đến một tay cầm ràng buộc khác hoặc đến ranh giới của phần tử cha. Một đường gấp khúc sẽ thể hiện ràng buộc.

- Tay cầm thay đổi kích thước: Để thay đổi kích thước của phần tử, kéo các tay cầm thay đổi kích thước hình vuông. Khi kéo, tay cầm sẽ chuyển thành một góc xiên.

2.2 Thêm một Button vào bố cục

Khi được bật, công cụ **Autoconnect** sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử UI với bố cục cha. Sau khi bạn kéo phần tử vào bố cục, nó sẽ tạo ràng buộc dựa trên vị trí của phần tử.

Thực hiện các bước sau để thêm một **Button**:

1. Bắt đầu với một bố cục trống. Phần tử **TextView** không cần thiết, vì vậy khi nó vẫn đang được chọn, nhấn phím **Delete** hoặc chọn **Edit > Delete**. Bây giờ bạn có một bố cục hoàn toàn trống.

2. Kéo một **Button** từ bảng **Palette** vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** vào khu vực giữa phía trên của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo ràng buộc đến mép trên, bên trái và bên phải của bố cục như trong hình động bên dưới.

2.3 Add Button thứ 2 vào bố cục

1. Kéo một **Button** khác từ bảng **Palette** vào giữa bố cục như trong hình động bên dưới. **Autoconnect** có thể tự động tạo ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo chúng).

2. Kéo một ràng buộc dọc xuống mép dưới của bố cục (tham khảo hình bên dưới).

Bạn có thể xóa ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuột qua để hiển thị nút **Clear Constraints**. Nhấn vào nút này để xóa tất cả ràng buộc trên

phần tử đã chọn .Để xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm ràng buộc tương ứng.
Để xóa tất cả ràng buộc trong toàn bộ bố cục, nhấn vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này hữu ích nếu bạn muốn thiết lập lại toàn bộ ràng buộc trong bố cục của mình.

Bước 3 : Thay đổi thuộc tính của phần tử UI

Ngăn Thuộc tính (**Attributes**) cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng (UI). Bạn có thể tìm thấy các thuộc tính chung cho tất cả các View trong tài liệu của lớp View.

Trong nhiệm vụ này, bạn sẽ nhập các giá trị mới và thay đổi các giá trị của các thuộc tính quan trọng của Button, những thuộc tính này cũng áp dụng cho hầu hết các loại View.

3.1 Thay đổi kích cỡ của Button

Trình chỉnh sửa bố cục cung cấp các tay nắm thay đổi kích thước ở bốn góc của một View, giúp bạn có thể nhanh chóng điều chỉnh kích thước của nó. Bạn có thể kéo các tay nắm ở mỗi góc của View để thay đổi kích thước, nhưng làm như vậy sẽ đặt cứng các kích thước chiều rộng và chiều cao. Tránh đặt kích thước cố định cho hầu hết các phần tử View, vì các kích thước cố định không thể thích ứng với nội dung và các kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn Thuộc tính (Attributes) ở phía bên phải của trình chỉnh sửa bố cục để chọn một chế độ kích thước không sử dụng kích thước cố định. Ngăn Thuộc tính bao gồm một bảng điều khiển kích thước hình vuông, được gọi là view inspector, ở phía trên. Các biểu tượng bên trong hình vuông biểu thị các cài đặt chiều cao và chiều rộng như sau:

Trong hình trên:

1. **Điều khiển chiều cao**: Điều khiển này xác định thuộc tính layout_height và xuất hiện dưới dạng hai đoạn ở phía trên và dưới của hình vuông. Các góc nghiêng cho thấy rằng điều khiển này được đặt thành wrap_content, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần thiết để phù hợp với nội dung của nó. Số "8" biểu thị một lề tiêu chuẩn được đặt là 8dp.

2. **Điều khiển chiều rộng**: Điều khiển này xác định thuộc tính layout_width và xuất hiện dưới dạng hai đoạn ở phía bên trái và bên phải của hình vuông. Các góc nghiêng cho thấy rằng điều khiển này được đặt thành wrap_content, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần thiết để phù hợp với nội dung của nó, với lề tối đa là 8dp.

3. **Nút đóng ngăn Thuộc tính**: Nhấp vào để đóng ngăn này.

Thực hiện các bước sau:

1. Chọn **nút (Button) trên cùng** trong ngăn **Component Tree**.

2. Nhấp vào **tab Attributes** ở phía bên phải của cửa sổ **layout editor**.

3. Nhấp **hai lần** vào điều khiển chiều rộng:

- Lần nhấp đầu tiên thay đổi thành **Fixed** (đường thẳng).

- Lần nhấp thứ hai thay đổi thành **Match Constraints** (lò xo cuộn), như trong hình động bên dưới.

Khi thay đổi điều khiển chiều rộng, thuộc tính layout_width trong **ngăn Attributes** hiển thị giá trị **match_constraint**, và phần tử **Button** sẽ kéo giãn theo chiều ngang để lấp đầy khoảng trống giữa hai bên của layout.

4. Chọn **nút (Button) thứ hai**, và thực hiện thay đổi layout_width tương tự như bước trước đó, như trong hình minh họa bên dưới.

Như đã trình bày ở các bước trước, các thuộc tính layout_width và layout_height trong ngăn **Attributes** sẽ thay đổi khi bạn điều chỉnh các điều khiển chiều rộng và chiều cao trong **inspector**. Các thuộc tính này có thể nhận một trong ba giá trị sau trong **ConstraintLayout**:

- **match_constraint**: Mở rộng phần tử **View** để lấp đầy **parent** theo chiều rộng hoặc chiều cao—tối đa đến **margin** nếu được đặt. Trong trường hợp này, **parent** chính là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.

- **wrap_content**: Thu nhỏ kích thước của phần tử **View** sao cho nó vừa đủ để bao bọc nội dung bên trong. Nếu không có nội dung, **View** sẽ trở nên **vô hình**.

- **Fixed size (kích thước cố định)**: Để đặt kích thước cố định mà vẫn có thể thích ứng với kích thước màn hình của thiết bị, hãy sử dụng đơn vị **density-independent pixels (dp)**. Ví dụ: 16dp có nghĩa là **16 pixel độc lập với mật độ màn hình**.

Mẹo: Nếu bạn thay đổi thuộc tính layout_width bằng menu bật lên của nó, giá trị của layout_width sẽ được đặt thành **0** vì không có kích thước cố định nào được thiết lập. Thiết lập này tương đương với **match_constraint**—phần tử có thể mở rộng tối đa để đáp ứng các ràng buộc và khoảng lề đã đặt.

3.2 Thay đổi thuộc tính của Button

Để xác định duy nhất từng View trong một Activity layout, mỗi View hoặc lớp con của View (chẳng hạn như Button) cần có một ID duy nhất. Ngoài ra, các phần tử Button cần có văn bản để sử dụng được. Các phần tử View cũng có thể có nền là màu sắc hoặc hình ảnh.

Bảng thuộc tính cung cấp quyền truy cập vào tất cả các thuộc tính có thể gán cho một phần tử View. Bạn có thể nhập giá trị cho từng thuộc tính, chẳng hạn như android:id, background, textColor và text.

Hình động sau đây minh họa cách thực hiện các bước này:

1. Sau khi chọn Button đầu tiên, chỉnh sửa trường ID ở đầu Bảng thuộc tính thành **button_toast** cho thuộc tính android:id, dùng để xác định phần tử trong bố cục.
2. Đặt thuộc tính background thành **@color/colorPrimary**. (Khi bạn nhập **@c**, các tùy chọn sẽ xuất hiện để dễ dàng lựa chọn.)
3. Đặt thuộc tính textColor thành **@android:color/white.**
4. Chỉnh sửa thuộc tính text thành T**oast**.

5. Thực hiện các thay đổi thuộc tính tương tự cho **Button** thứ hai, sử dụng button_count làm **ID**, đặt thuộc tính text thành **Count**, và sử dụng cùng màu cho background và text như các bước trước.

Bước 4: Thêm TextEdit và đặt thuộc tính cho nó

Một trong những lợi ích của **ConstraintLayout** là khả năng căn chỉnh hoặc ràng buộc các phần tử so với các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một TextView vào giữa bố cục và ràng buộc nó theo chiều ngang với các lề và theo chiều dọc với hai Button. Sau đó, bạn sẽ thay đổi các thuộc tính cho TextView trong **Attributes pane.**

4.1 Thêm TextView và ràng buộc

1. Như hình động bên dưới, kéo một **TextView** từ ngăn **Palette** vào phần trên của bố cục, và kéo một ràng buộc từ đỉnh của **TextView** đến tay cầm ở phía dưới của **Button Toast**. Việc này sẽ ràng buộc **TextView** nằm bên dưới **Button**.

2. Như hình động bên dưới, kéo một ràng buộc từ đáy của **TextView** đến tay cầm ở phía trên của **Button Count**, và từ hai bên của **TextView** đến hai bên của bố cục. Việc này sẽ ràng buộc **TextView** nằm giữa bố cục, giữa hai **Button**.

4.2 Thiết lập thuộc tính cho TextView

Với **TextView** được chọn, mở ngăn **Attributes** nếu nó chưa được mở. Thiết lập các thuộc tính cho **TextView** như trong hình động bên dưới. Các thuộc tính mà bạn chưa gặp sẽ được giải thích sau hình

1. Đặt **ID** thành show_count.
2. Đặt **text** thành 0.
3. Đặt **textSize** thành 160sp.
4. Đặt **textStyle** thành **B** (đậm) và **textAlignment** thành ALIGNCENTER (căn giữa đoạn văn bản).
5. Thay đổi điều khiển kích thước ngang và dọc (**layout_width** và **layout_height**) thành match_constraint.
6. Đặt **textColor** thành @color/colorPrimary.
7. Cuộn xuống ngăn **Attributes**, nhấp vào **View all attributes**, cuộn xuống trang thứ hai đến **background**, và nhập #FFF00 để đặt màu vàng nhạt.
8. Cuộn xuống **gravity**, mở rộng mục này và chọn center_ver (căn giữa theo chiều dọc).

- **textSize**: Kích thước văn bản của **TextView**. Trong bài học này, kích thước được đặt thành 160sp. **sp** (scale-independent pixel) là đơn vị tỷ lệ theo mật độ màn hình và cài đặt kích thước phông chữ của người dùng. Khi chỉ định kích thước phông chữ, nên sử dụng đơn vị **sp** để đảm bảo văn bản điều chỉnh phù hợp với cả mật độ màn hình và tùy chọn của người dùng.
- **textStyle** và **textAlignment**: **textStyle** được đặt thành **B** (đậm) và **textAlignment** được đặt thành ALIGNCENTER (căn giữa đoạn văn bản).
- **gravity**: Thuộc tính **gravity** xác định cách một **View** được căn chỉnh trong **View** hoặc **ViewGroup** cha. Trong bước này, **TextView** được căn giữa theo chiều dọc trong **ConstraintLayout**.

Bạn có thể nhận thấy rằng thuộc tính **background** nằm ở trang đầu của ngăn **Attributes** đối với **Button**, nhưng lại nằm ở trang thứ hai đối với **TextView**. Ngăn **Attributes** thay đổi tùy theo loại **View**: Các thuộc tính phổ biến nhất của loại **View** sẽ xuất hiện trên trang đầu tiên, trong khi các thuộc tính khác được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của ngăn **Attributes**, hãy nhấp vào biểu tượng trong thanh công cụ ở đầu ngăn.

Bước 5: Chỉnh bố cục trong XML

Bố cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử giao diện người dùng trong Component Tree. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hiển thị bên dưới. Cùng một cảnh báo xuất hiện cho cả ba phần tử: các chuỗi được mã hóa cứng (hardcoded strings) nên sử dụng tài nguyên.

Cách dễ nhất để khắc phục sự cố bố cục là chỉnh sửa bố cục trong XML. Mặc dù trình chỉnh sửa bố cục là một công cụ mạnh mẽ, nhưng một số thay đổi sẽ dễ thực hiện hơn khi chỉnh sửa trực tiếp trong mã nguồn XML.

5.1 Mở mã XML cho bố cục

Đối với nhiệm vụ này, mở tệp **activity_main.xml** nếu nó chưa được mở và nhấp vào tab **Text** ở dưới cùng của trình chỉnh sửa bố cục.

Trình chỉnh sửa XML xuất hiện, thay thế các ngăn thiết kế và bản vẽ. Như bạn có thể thấy trong hình dưới đây, hiển thị một phần mã XML của bố cục, các cảnh báo được đánh dấu—các chuỗi mã cứng **"Toast"** và **"Count"**. (Chuỗi mã cứng **"0"** cũng được đánh dấu nhưng không hiển thị trong hình.) Di chuột qua chuỗi mã cứng **"Toast"** để xem thông báo cảnh báo.

```
activity_main.xml   MainActivity.java   build.gradle.kts (HelloWorld   build.gradle.kts (:app   proguard-ru
2   <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/
27      <Button
28          android:id="@+id/button_count"
29          android:layout_width="0dp"
30          android:layout_height="wrap_content"
31          android:layout_marginStart="8dp"
32          android:layout_marginTop="8dp"
33          android:layout_marginEnd="8dp"
34          android:layout_marginBottom="8dp"
35          android:background="?attr/colorPrimary"
36          android:text="Count"
37          android:textColor="@android:color/white"
38          app:layout_constraintBottom_toBottomOf="parent"
39          app:layout_constraintEnd_toEndOf="parent"
40          app:layout_constraintHorizontal_bias="0.0"
41          app:layout_constraintStart_toStartOf="parent"
42          app:layout_constraintTop_toTopOf="parent"
43          app:layout_constraintVertical_bias="0.923" />
44
45      <Button
46          android:id="@+id/button_toast"
47          android:layout_width="0dp"
48          android:layout_height="wrap_content"
49          android:layout_marginStart="8dp"
50          android:layout_marginTop="8dp"
51          android:layout_marginEnd="8dp"
52          android:layout_marginBottom="8dp"
53          android:background="?attr/colorPrimary"
54          android:text="Toast"
55          android:textColor
56          app:layout_constr
57          app:layout_constr
58          app:layout_constr
59          app:layout_constraintStart_toStartOf="parent"
60          app:layout_constraintTop_toTopOf="parent"

Hardcoded string "Toast", should use @string resource Toggle info (Ctrl+F1)
Extract string resource  Alt+Shift+Enter   More actions...  Alt+Enter
```

5.2 Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng chuỗi, một phương pháp hay nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Việc đặt các chuỗi trong một tệp riêng biệt giúp dễ dàng quản lý chúng hơn, đặc biệt nếu bạn sử dụng những chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và địa phương hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhấp một lần vào từ "Toast" (cảnh báo được tô sáng đầu tiên).
2. Nhấn **Alt-Enter** trên Windows hoặc **Option-Enter** trên macOS, sau đó chọn **Extract string resource** từ menu bật lên.
3. Nhập **button_label_toast** cho **Resource name**.

4. Nhấn **OK**. Một tài nguyên chuỗi được tạo trong tệp **res/values/strings.xml**, và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: @string/button_label_toast.
5. Trích xuất các chuỗi còn lại:

- **button_label_count** cho "Count"
- **count_initial_value** cho "0"

6. Trong **Project > Android**, mở rộng **values** trong **res**, sau đó nhấp đúp vào **strings.xml** để xem các tài nguyên chuỗi trong tệp **strings.xml**.

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_toast">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
</resources>
```

7. Bạn cần một chuỗi khác để sử dụng trong nhiệm vụ tiếp theo nhằm hiển thị một thông báo. Thêm vào tệp **strings.xml** một tài nguyên chuỗi mới có tên **toast_message** với nội dung "Hello Toast!":

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_toast">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
    <string name="toast_message">Hello Toast!</string>
</resources>
```

Mẹo: Các tài nguyên chuỗi bao gồm tên ứng dụng, tên này sẽ xuất hiện trên thanh ứng dụng ở đầu màn hình nếu bạn khởi tạo dự án ứng dụng bằng mẫu **Empty Template**. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên **app_name**.

Bước 6: Thêm trình xử lý onClick cho các nút

Trong nhiệm vụ này, bạn thêm một phương thức Java cho mỗi nút trong **MainActivity**, phương thức này sẽ được thực thi khi người dùng nhấn vào nút.

6.1 Thêm thuộc tính onClick và trình xử lý cho từng Button

Trình xử lý sự kiện nhấp chuột là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường **onClick** của **Attributes pane** ở tab **Design**. Bạn cũng có thể chỉ định tên của phương thức xử lý trong **trình chỉnh sửa XML** bằng cách thêm thuộc tính android:onClick vào nút **Button**. Bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý, và trình chỉnh sửa XML cung cấp một cách tự động để tạo các phương thức đó.

1. Khi trình chỉnh sửa XML đang mở (tab **Text**), tìm **Button** có thuộc tính android:id được đặt thành button_toast.
2. Thêm thuộc tính android:onClick vào cuối phần tử button_toast, sau thuộc tính cuối cùng và trước dấu kết thúc />.
3. Nhấp vào biểu tượng **bóng đèn đỏ** xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity**, rồi nhấp **OK**.

- Nếu không thấy biểu tượng **bóng đèn đỏ**, nhấp vào tên phương thức ("showToast"), nhấn Alt+Enter (hoặc Option+Enter trên Mac), chọn **Create 'showToast(view)' in MainActivity**, rồi nhấp **OK**.

- Hành động này sẽ tạo một phương thức xử lý sự kiện (showToast()) trong **MainActivity**.

4. Lặp lại các bước trên cho **Button** có android:id="button_count":

- Thêm thuộc tính android:onClick.

- Tạo phương thức xử lý sự kiện.

Mã XML cho các phần tử giao diện người dùng trong ConstraintLayout bây giờ trông như thế này:

```
<Button
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="?attr/colorPrimary"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.923"
    android:onClick="showCount"/>

<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="?attr/colorPrimary"
    android:text="@string/button_label_toast"
    android:textColor="@android:color/white"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.142"
    android:onClick="showToast" />
```

5. Nếu **MainActivity.java** chưa mở, hãy mở **Project > Android**, mở rộng thư mục java, sau đó mở rộng com.example.android.hellotoast, và nhấp đúp vào **MainActivity.java** để mở trình chỉnh sửa mã.

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        Log.d( tag: "MainActivity",  msg: "Hello World");
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }

    1 usage
    public void showToast(View view) {
    }

    1 usage
    public void showCount(View view) {
    }
}
```

6.2 Chỉnh sửa trình xử lý Button Toast

Bây giờ bạn sẽ chỉnh sửa phương thức showToast()—trình xử lý sự kiện nhấn nút Toast trong MainActivity—để hiển thị một thông báo. Toast cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ, chỉ chiếm không gian cần thiết cho nội dung thông báo. Hoạt động hiện tại vẫn hiển thị và có thể tương tác. Toast hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn—bạn có thể thêm một thông báo Toast để hiển thị kết quả khi nhấn một Button hoặc thực hiện một hành động nào đó.

Thực hiện các bước sau để chỉnh sửa trình xử lý sự kiện của Button Toast:

1. Xác định phương thức showToast() vừa được tạo.

```java
1 usage
public void showToast(View view) {
}
```

2. Tạo một đối tượng Toast bằng cách gọi phương thức makeText() của lớp Toast.

3. Cung cấp ngữ cảnh (context) của Activity. Vì Toast hiển thị trên giao diện của Activity, hệ thống cần thông tin về Activity hiện tại. Khi đang ở trong Activity cần sử dụng context, bạn có thể dùng this làm shortcut.
4. Cung cấp thông báo cần hiển thị, chẳng hạn như một chuỗi tài nguyên (string resource) toast_message mà bạn đã tạo ở bước trước. Chuỗi tài nguyên toast_message được xác định bằng R.string.toast_message.
5. Cung cấp thời gian hiển thị. Ví dụ, Toast.LENGTH_SHORT hiển thị Toast trong thời gian ngắn.
    - Toast.LENGTH_LONG: Hiển thị khoảng 3,5 giây.
    - Toast.LENGTH_SHORT: Hiển thị khoảng 2 giây.
6. Hiển thị Toast bằng cách gọi show().

```
1 usage
public void showToast(View view) {
    Toast toast = Toast.makeText( context: this, R.string.toast_message, Toast.LENGTH_SHORT);
    toast.show();
}
```

Sau khi hoàn thành, hãy chạy ứng dụng và kiểm tra xem thông báo Toast có xuất hiện khi nhấn Button Toast hay không.

07:27
100%
Hello Toast
Toast
0
Hello Toast!

6.3 Chỉnh sửa trình xử lý sự kiện của Button Count

Bây giờ bạn sẽ chỉnh sửa phương thức countUp()—trình xử lý sự kiện của nút Count trong MainActivity—để hiển thị số đếm hiện tại sau khi nhấn Count. Mỗi lần nhấn sẽ tăng số đếm lên một đơn vị.

Mã cho trình xử lý phải:

- Theo dõi số đếm khi nó thay đổi.
- Gửi số đếm đã cập nhật đến TextView để hiển thị.

Thực hiện các bước sau để chỉnh sửa trình xử lý nút Count:

1. Xác định phương thức countUp() vừa được tạo.
2. Để theo dõi số đếm, bạn cần một biến thành viên riêng tư. Mỗi lần nhấn nút Count sẽ tăng giá trị của biến này. Nhập đoạn mã sau, đoạn mã này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn đỏ:

   ```
   public void countUp(View view) {
       mCount++;
   }
   ```

   Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy chọn biểu thức mCount++. Biểu tượng bóng đèn cuối cùng sẽ xuất hiện.

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Điều này tạo một biến thành viên riêng tư ở đầu MainActivity, và Android Studio giả định rằng bạn muốn nó là một số nguyên (int).
4. Thay đổi câu lệnh biến thành viên riêng tư để khởi tạo biến với giá trị bằng 0.
5. Cùng với biến trên, bạn cũng cần một biến thành viên riêng tư để tham chiếu đến TextView có ID show_count, biến này sẽ được thêm vào trình xử lý nhấp chuột. Đặt tên biến là mShowCount.
6. Bây giờ bạn đã có mShowCount, bạn có thể lấy tham chiếu đến TextView bằng cách sử dụng ID bạn đã đặt trong tệp bố cục. Để lấy tham chiếu này chỉ một lần, hãy xác định nó trong phương thức onCreate(). Như bạn sẽ học trong một bài học khác, phương thức onCreate() được sử dụng để nạp bố cục, có nghĩa là thiết lập giao diện màn hình thành bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các thành phần giao diện khác trong bố cục, chẳng hạn như TextView.

7. Thêm câu lệnh findViewById vào cuối phương thức:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);
    mShowCount = (TextView) findViewById(R.id.show_count);A
```

   Một View, giống như một chuỗi, là một tài nguyên có thể có ID. Lệnh findViewById nhận ID của một View làm tham số và trả về View. Vì phương thức này trả về một View, bạn phải ép kiểu kết quả về kiểu TextView.

8. Sau khi đã gán TextView vào biến mShowCount, bạn có thể sử dụng biến này để đặt văn bản trong TextView thành giá trị của biến mCount. Thêm đoạn mã sau vào phương thức countUp().
9. Chạy ứng dụng để kiểm tra xem số đếm có tăng khi bạn nhấn nút Count hay không.

08:16
100%
Hello Toast
Toast
10
Count

Tổng kết

View, ViewGroup và Layouts:

- Tất cả các phần tử giao diện người dùng (UI elements) đều là lớp con của lớp View và do đó kế thừa nhiều thuộc tính của lớp View cha.
- Các phần tử View có thể được nhóm lại bên trong một ViewGroup, hoạt động như một vùng chứa. Mối quan hệ này là cha - con, trong đó cha là ViewGroup, còn con là View hoặc một ViewGroup khác.
- Phương thức onCreate() được sử dụng để nạp bố cục (inflate layout), nghĩa là thiết lập nội dung màn hình thành bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong bố cục.
- Một View, giống như một chuỗi, là một tài nguyên có thể có id. Lệnh findViewById nhận ID của một View làm tham số và trả về đối tượng View tương ứng.

Sử dụng trình chỉnh sửa layout:

- Nhấp vào tab **Design** để thao tác với các phần tử và bố cục, hoặc tab **Text** để chỉnh sửa mã XML của bố cục.
- Trong tab **Design**, **Palettes** pane hiển thị các phần tử giao diện có thể sử dụng trong bố cục của ứng dụng. **Component tree** pane hiển thị cấu trúc phân cấp của các phần tử giao diện.
- Design pane và Blueprint pane của trình chỉnh sửa bố cục hiển thị các phần tử giao diện có trong bố cục.
- **Attributes tab** hiển thị bảng **Attributes** để thiết lập thuộc tính cho một phần tử giao diện.
- Constraint handle: Nhấp vào chấm tròn ở mỗi cạnh của một phần tử và kéo đến một chấm tròn khác hoặc ranh giới của ViewGroup cha để tạo ràng buộc (constraint). Ràng buộc này được biểu diễn bằng đường gấp khúc.
- Resizing handle: Kéo các ô vuông để thay đổi kích thước phần tử. Khi kéo, tay cầm thay đổi thành một góc xiên.
- Autoconnect tool (công cụ kết nối tự động) tự động tạo hai hoặc nhiều ràng buộc cho một phần tử giao diện với bố cục cha dựa trên vị trí của nó.
- Xóa ràng buộc: Chọn phần tử, di chuột qua để hiển thị nút Clear Constraints, nhấp vào để xóa tất cả ràng buộc. Để xóa một ràng buộc cụ thể, nhấp vào chấm tròn của ràng buộc đó.

- **Attributes pane** chứa tất cả thuộc tính XML có thể áp dụng cho một phần tử giao diện. Nó cũng có **View Inspector**, một bảng điều chỉnh kích thước vuông nhỏ ở đầu. Các biểu tượng bên trong biểu thị các cài đặt về chiều rộng và chiều cao.

Cài đặt chiều rộng và chiều cao của bố cục:

Các thuộc tính layout_width và layout_height thay đổi theo điều chỉnh kích thước trong View Inspector. Với ConstraintLayout, có ba giá trị có thể sử dụng:

- match_constraint: Mở rộng phần tử để lấp đầy vùng chứa theo chiều rộng hoặc chiều cao, giới hạn bởi phần margin (nếu có).
- wrap_content: Thu nhỏ kích thước phần tử sao cho vừa với nội dung của nó. Nếu không có nội dung, phần tử trở nên vô hình.
- Giá trị số cố định (dp - density-independent pixels): Định kích thước cố định, điều chỉnh theo kích thước màn hình thiết bị.

Trích xuất chuỗi tài nguyên (string resources):

1. Nhấp vào chuỗi cần trích xuất, nhấn Alt + Enter (Option + Enter trên Mac) và chọn **Extract string resources** từ menu bật lên.
2. Đặt tên cho tài nguyên chuỗi (**Resource name**).
3. Nhấp **OK**. Một tài nguyên chuỗi được tạo trong tệp res/values/strings.xml, và chuỗi trong mã sẽ được thay thế bằng một **tham chiếu tài nguyên**: @string/button_label_toast.

Xử lý sự kiện nhấp:

- **Click handler** là một phương thức được gọi khi người dùng **nhấp hoặc chạm** vào một phần tử giao diện.
- Để chỉ định một click handler cho một phần tử giao diện (như Button), **Trong tab Design**, nhập tên phương thức vào trường **onClick** trong bảng **Attributes** hoặc trong XML , thêm thuộc tính android:onClick vào phần tử (Button).
- **Tạo phương thức click handler** trong MainActivity với tham số View. Ví dụ : public void showToast(View view) {/...}.
- Bạn có thể tìm thấy tất cả thuộc tính của Button trong tài liệu lớp **Button**, và thuộc tính của TextView trong tài liệu lớp **TextView**.

Hiển thị thông báo Toast:

Toast là một cách để hiển thị một thông báo nhỏ trong popup, chỉ chiếm không gian cần thiết cho nội dung. Các bước tạo Toast:

1. Gọi phương thức makeText() từ lớp Toast.
2. Cung cấp **context** của ứng dụng (Activity) và **chuỗi thông báo** (có thể là tài nguyên chuỗi).
3. Xác định **thời gian hiển thị** ,Toast.LENGTH_SHORT: Hiển thị trong thời gian ngắn. Toast.LENGTH_LONG: Hiển thị trong thời gian dài.
4. Gọi show() để hiển thị Toast.

1.3) Trình chỉnh sửa bố cục

Mở đầu

Như bạn đã học trong **Phần 1.2: Giao diện người dùng tương tác đầu tiên**, bạn có thể xây dựng giao diện người dùng (UI) bằng **ConstraintLayout** trong trình chỉnh sửa bố cục (layout editor), bố cục này sắp xếp các phần tử UI bằng cách sử dụng các ràng buộc (constraint) kết nối với các phần tử khác và với các cạnh của bố cục. **ConstraintLayout** được thiết kế để giúp bạn dễ dàng kéo các phần tử UI vào trình chỉnh sửa bố cục.

ConstraintLayout là một **ViewGroup**, một loại View đặc biệt có thể chứa các đối tượng View khác (gọi là **children** hoặc **child views**). Bài thực hành này giới thiệu thêm nhiều tính năng của ConstraintLayout và trình chỉnh sửa bố cục.

Bài thực hành này cũng giới thiệu hai lớp con khác của ViewGroup:

- **LinearLayout**: Một nhóm sắp xếp các phần tử View con bên trong nó theo chiều ngang hoặc chiều dọc.
- **RelativeLayout**: Một nhóm các phần tử View con, trong đó mỗi phần tử View được định vị và căn chỉnh tương đối với các phần tử khác trong ViewGroup. Vị trí của các phần tử View con được mô tả dựa trên mối quan hệ với nhau hoặc với ViewGroup cha.

Những gì bạn nên biết

Bạn nên biết:

- Tạo ứng dụng Hello World bằng Android Studio.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị thật.

- Tạo bố cục đơn giản cho ứng dụng bằng ConstraintLayout.
- Trích xuất và sử dụng tài nguyên chuỗi (string resources).

Những gì bạn sẽ học

- Cách tạo một biến thể bố cục cho chế độ ngang (landscape).
- Cách tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng ràng buộc đường cơ sở (baseline constraint) để căn chỉnh các phần tử giao diện người dùng với văn bản.
- Cách sử dụng các nút gói (pack) và căn chỉnh (align) để sắp xếp các phần tử trong bố cục.
- Cách định vị các View trong LinearLayout.
- Cách định vị các View trong RelativeLayout.

Những gì bạn sẽ làm

- Tạo một biến thể bố cục cho chế độ hiển thị ngang.
- Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Chỉnh sửa bố cục để thêm ràng buộc (constraints) cho các phần tử giao diện người dùng.
- Sử dụng ràng buộc đường cơ sở (baseline constraints) trong ConstraintLayout để căn chỉnh các phần tử với văn bản.
- Sử dụng các nút gói (pack) và căn chỉnh (align) trong ConstraintLayout để sắp xếp các phần tử.
- Thay đổi bố cục để sử dụng LinearLayout.
- Định vị các phần tử trong LinearLayout.
- Thay đổi bố cục để sử dụng RelativeLayout.
- Sắp xếp lại các View trong bố cục chính sao cho liên kết tương đối với nhau.

Tổng quan ứng dụng

Ứng dụng Hello Toast trong bài học trước sử dụng ConstraintLayout để sắp xếp các phần tử giao diện người dùng trong bố cục Activity, như hình minh họa bên dưới.

08:16
100%
Hello Toast
Toast
10
Count

Để thực hành nhiều hơn với ConstraintLayout, bạn sẽ tạo một biến thể của bố cục này cho chế độ ngang như trong hình minh họa bên dưới.

Bạn cũng sẽ học cách sử dụng ràng buộc đường cơ sở (baseline constraints) và một số tính năng căn chỉnh của ConstraintLayout bằng cách tạo một biến thể bố cục khác cho màn hình máy tính bảng.

Ngoài ra, bạn sẽ tìm hiểu về các lớp con khác của ViewGroup, chẳng hạn như LinearLayout và RelativeLayout, đồng thời thay đổi bố cục của ứng dụng Hello Toast để sử dụng chúng.

Bước 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng Hello Toast để phù hợp với chế độ hiển thị ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể của bố cục cho điện thoại theo chiều ngang (còn gọi là landscape) và chiều dọc (còn gọi là portrait), cũng như cho các màn hình lớn hơn như máy tính bảng.

Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trong hai thanh công cụ trên cùng của trình chỉnh sửa bố cục. Thanh công cụ trên cùng cho phép bạn cấu hình giao diện của bản xem trước bố cục trong trình chỉnh sửa.

Trong hình minh họa trên:

1. **Chọn Bề mặt Thiết kế**: Chọn *Design* để hiển thị bản xem trước có màu của bố cục hoặc *Blueprint* để chỉ hiển thị đường viền của từng phần tử giao diện người dùng. Để xem cả hai ngăn cạnh nhau, chọn *Design* + *Blueprint*.

2. **Hướng trong Trình chỉnh sửa**: Chọn *Portrait* hoặc *Landscape* để hiển thị bản xem trước theo chiều dọc hoặc ngang. Điều này hữu ích để xem trước bố cục mà không cần chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo các bố cục thay thế, chọn *Create Landscape Variation* hoặc các biến thể khác.

3. **Thiết bị trong Trình chỉnh sửa**: Chọn loại thiết bị (*điện thoại/máy tính bảng, Android TV hoặc Android Wear*).

4. **Phiên bản API trong Trình chỉnh sửa**: Chọn phiên bản Android để hiển thị bản xem trước.

5. **Chủ đề trong Trình chỉnh sửa**: Chọn một chủ đề (*chẳng hạn như AppTheme*) để áp dụng cho bản xem trước.

6. **Ngôn ngữ trong Trình chỉnh sửa**: Chọn ngôn ngữ và khu vực hiển thị cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có sẵn trong tài nguyên chuỗi. Bạn cũng có thể chọn *Preview as Right To Left* để xem bố cục như thể đã chọn một ngôn ngữ RTL.

Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các phần tử UI trong *ConstraintLayout* và thu phóng hoặc di chuyển bản xem trước.

Trong hình trên:

1. **Show**: Chọn *Show Constraints* và *Show Margins* để hiển thị hoặc ẩn chúng trong bản xem trước.
2. **Autoconnect**: Bật hoặc tắt *Autoconnect*. Khi bật, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như *Button*) đến bất kỳ vị trí nào trong bố cục để tự động tạo ràng buộc với bố cục cha.
3. **Clear All Constraints**: Xóa tất cả các ràng buộc trong toàn bộ bố cục.
4. **Infer Constraints**: Tạo ràng buộc bằng cách suy luận.
5. **Default Margins**: Đặt lề mặc định.
6. **Pack**: Gom nhóm hoặc mở rộng các phần tử đã chọn.
7. **Align**: Căn chỉnh các phần tử đã chọn.
8. **Guidelines**: Thêm đường hướng dẫn dọc hoặc ngang.
9. **Zoom/pan controls:** Thu phóng hoặc di chuyển.

Mẹo: Để tìm hiểu thêm về cách sử dụng *layout editor*, hãy xem **Build a UI with Layout Editor**. Để tìm hiểu thêm về cách xây dựng bố cục với *ConstraintLayout*, hãy xem **Build a Responsive UI with ConstraintLayout**.

1.1 Xem trước bố cục trong chế độ ngang

Để xem trước bố cục của ứng dụng Hello Toast trong chế độ ngang, hãy làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ bài học trước. **Lưu ý:** Nếu bạn đã tải xuống mã nguồn giải pháp của HelloToast, bạn cần xóa các bố cục landscape và extra-large đã hoàn thành mà bạn sẽ tạo trong nhiệm vụ này.Chuyển từ **Project > Android** sang **Project > Project Files** trong bảng Project. Mở rộng đường dẫn: app > src/main > res, chọn cả hai thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**. Sau đó, chuyển bảng Project trở lại **Project > Android**.
2. Mở tệp bố cục activity_main.xml. Nhấp vào tab **Design** nếu nó chưa được chọn.
3. Nhấp vào nút **Orientation in Editor** trên thanh công cụ phía trên.
4. Chọn **Switch to Landscape** trong menu thả xuống. Bố cục sẽ hiển thị trong chế độ ngang như hình dưới đây. Để quay lại chế độ dọc, chọn **Switch to Portrait**

1.2 Tạo một biến thể bố cục cho chế độ ngang

Sự khác biệt về mặt hiển thị giữa chế độ dọc và chế độ ngang trong bố cục này là chữ số (0) trong phần tử TextView show_count nằm quá thấp trong chế độ ngang—quá gần với nút Count. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử TextView có thể hiển thị quá lớn hoặc không căn giữa vì kích thước văn bản được cố định ở 160sp.

Để khắc phục vấn đề này trong chế độ ngang mà không ảnh hưởng đến chế độ dọc, bạn có thể tạo một biến thể của bố cục ứng dụng Hello Toast dành riêng cho chế độ ngang. Hãy làm theo các bước sau:

1. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
2. Chọn **Create Landscape Variation**. Một cửa sổ chỉnh sửa mới sẽ mở ra với tab land/activity_main.xml, hiển thị bố cục dành cho chế độ ngang (horizontal). Bạn có thể chỉnh sửa bố cục này dành riêng cho chế độ ngang mà không ảnh hưởng đến bố cục gốc ở chế độ dọc (vertical).
3. Trong ngăn **Project > Android**, mở thư mục res > layout, bạn sẽ thấy Android Studio đã tự động tạo một biến thể có tên activity_main.xml (land).

1.3 Xem trước bố cục trên các thiết bị khác nhau

Bạn có thể xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thực hoặc trình giả lập. Thực hiện các bước sau:

1. Tab **land/activity_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục; nếu không, hãy nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục **layout**.
2. Nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng.

3. Chọn một thiết bị khác trong menu thả xuống. Ví dụ, chọn **Nexus 4**, **Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong các bản xem trước. Những khác biệt này là do kích thước văn bản cố định trong **TextView**.

1.4 Thay đổi bố cục cho chế độ ngang

Bạn có thể sử dụng ngăn **Attributes** trong tab **Design** để thiết lập hoặc thay đổi thuộc tính, nhưng đôi khi chỉnh sửa trực tiếp mã XML trong tab **Text** sẽ nhanh hơn. Tab **Text** hiển thị mã XML và cung cấp tab **Preview** ở bên phải cửa sổ để xem trước bố cục, như minh họa trong hình dưới đây.

Để thay đổi bố cục, hãy làm theo các bước sau:

1. Tab **land/activity_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục; nếu không, hãy nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục **layout**.
2. Nhấp vào tab **Text** và tab **Preview** (nếu chưa được chọn).
3. Tìm phần tử **TextView** trong mã XML.

4. Thay đổi thuộc tính android:textSize="160sp" thành android:textSize="120sp". Bản xem trước bố cục sẽ hiển thị kết quả:

5. Chọn các thiết bị khác nhau trong menu thả xuống **Device in Editor** để xem bố cục hiển thị như thế nào trên các thiết bị khác nhau ở chế độ ngang. Trong ngăn chỉnh sửa, tab **land/activity_main.xml** hiển thị bố cục cho chế độ ngang. Tab **activity_main.xml** hiển thị bố cục chưa thay đổi cho chế độ dọc.Bạn có thể chuyển qua lại bằng cách nhấp vào các tab.

6. Chạy ứng dụng trên trình giả lập hoặc thiết bị thật, sau đó chuyển đổi giữa chế độ dọc và ngang để xem cả hai bố cục.

1.5 Tạo một biến thể bố cục cho máy tính bảng

Như bạn đã học trước đó, bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng. Nếu bạn chọn một thiết bị như **Nexus 10** (một máy tính bảng) từ menu, bạn sẽ thấy rằng bố cục không phù hợp với màn hình máy tính bảng—văn bản trên mỗi nút **Button** quá nhỏ và cách sắp xếp các nút ở trên cùng và dưới cùng không lý tưởng cho một màn hình lớn.

Để khắc phục điều này cho máy tính bảng trong khi vẫn giữ nguyên bố cục cho điện thoại ở cả chế độ dọc và ngang, bạn có thể tạo một biến thể bố cục hoàn toàn khác dành riêng cho máy tính bảng. Hãy làm theo các bước sau:

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị các bảng thiết kế và sơ đồ.
2. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
3. Chọn **Create layout x-large Variation**.

Một cửa sổ chỉnh sửa mới mở ra với tab **xlarge/activity_main.xml**, hiển thị bố cục cho thiết bị có kích thước màn hình máy tính bảng. Trình chỉnh sửa cũng tự động chọn một thiết bị máy tính bảng, chẳng hạn như Nexus 9 hoặc Nexus 10, để hiển thị bản xem trước. Bạn có thể chỉnh sửa bố cục này dành riêng cho máy tính bảng mà không ảnh hưởng đến các bố cục khác.

1.6 Thay đổi biến thể bố cục cho máy tính bảng

Bạn có thể sử dụng ngăn Attributes trong tab **Design** để thay đổi thuộc tính cho bố cục này.

1. Tắt công cụ Autoconnect trên thanh công cụ. Đảm bảo rằng công cụ này đã bị vô hiệu hóa.
2. Xóa tất cả ràng buộc trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ. Khi đã xóa ràng buộc, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trên bố cục.
3. Thay đổi kích thước phần tử , trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở bốn góc của mỗi phần tử. Trong **Component Tree**, chọn TextView có ID show_count. Để dễ dàng kéo các nút (Button), hãy kéo một góc của TextView để thay đổi kích thước của nó, như minh họa trong hình động bên dưới.

Việc thay đổi kích thước một phần tử sẽ gán cứng (hardcode) các giá trị chiều rộng và chiều cao. Tránh gán cứng kích thước cho hầu hết các phần tử, vì bạn không thể dự đoán chúng sẽ hiển thị như thế nào trên các màn hình có kích thước và mật độ khác nhau. Bạn chỉ đang làm điều này để tạm thời di chuyển phần tử, và bạn sẽ điều chỉnh kích thước ở bước tiếp theo.

4. Chọn button_toast trong **Component Tree**, nhấp vào tab **Attributes** để mở bảng thuộc tính, sau đó thay đổi textSize thành **60sp** (#1 trong hình dưới). Thay đổi layout_width thành wrap_content (#2 trong hình dưới).

Như hiển thị ở phía bên phải của hình trên (2), bạn có thể nhấp vào bộ điều khiển chiều rộng của trình kiểm tra chế độ xem, xuất hiện dưới dạng hai đoạn ở bên trái và bên

phải của ô vuông, cho đến khi nó hiển thị Wrap Content. Ngoài ra, bạn cũng có thể chọn **wrap_content** từ menu layout_width.

Bạn sử dụng **wrap_content** để đảm bảo rằng nếu văn bản của nút được dịch sang một ngôn ngữ khác, nút sẽ tự động mở rộng hoặc thu hẹp để phù hợp với từ trong ngôn ngữ đó.

5. Chọn nút button_count trong **Component Tree**, thay đổi textSize thành **60sp** và layout_width thành wrap_content, sau đó kéo nút này lên phía trên **TextView** vào một khoảng trống trong bố cục.

1.7 Sử dụng ràng buộc đường cơ sở

Bạn có thể căn chỉnh một phần tử giao diện người dùng chứa văn bản, chẳng hạn như TextView hoặc Button, với một phần tử giao diện người dùng khác cũng chứa văn bản. Ràng buộc đường cơ sở cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở của văn bản khớp nhau.

1. Ràng buộc nút button_toast vào cạnh trên và cạnh trái của bố cục, kéo nút button_count đến một vị trí gần button_toast, sau đó ràng buộc button_count vào cạnh trái của button_toast, như được minh họa trong hình động.

2. Sử dụng ràng buộc đường cơ sở, bạn có thể ràng buộc button_count sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của button_toast. Chọn phần tử button_count, sau đó di chuột qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.

3. Nhấp vào nút ràng buộc đường cơ sở. Tay cầm đường cơ sở sẽ xuất hiện và nhấp nháy màu xanh lục như trong hình động. Nhấp và kéo đường ràng buộc đường cơ sở đến đường cơ sở của phần tử button_toast.

1.8 Mở rộng các nút theo chiều ngang

Nút gói (pack button) trên thanh công cụ cung cấp các tùy chọn để gom nhóm hoặc mở rộng các phần tử giao diện người dùng được chọn. Bạn có thể sử dụng nó để sắp xếp đều các nút Button theo chiều ngang trên bố cục.

1. Chọn nút button_count trong **Component Tree**, sau đó nhấn Shift và chọn nút button_toast để chọn cả hai.
2. Nhấp vào nút gói (pack button) trên thanh công cụ và chọn **Expand Horizontally**, như minh họa trong hình dưới đây.

Các phần tử Button mở rộng theo chiều ngang để lấp đầy bố cục như minh họa bên dưới.

3. Để hoàn thiện bố cục, ràng buộc show_count TextView vào cạnh dưới của button_toast và vào hai bên cũng như cạnh dưới của bố cục, như trong hình động bên dưới.

4. Bước cuối cùng là thay đổi layout_width và layout_height của show_count **TextView** thành **Match Constraints** và đặt textSize thành **200sp**. Bố cục cuối cùng trông giống như hình bên dưới.

5. Nhấp vào nút **Orientation in Editor** trên thanh công cụ và chọn **Switch to Landscape**. Bố cục dành cho máy tính bảng sẽ xuất hiện ở chế độ ngang như hình dưới. (Bạn có thể chọn **Switch to Portrait** để quay lại chế độ dọc.)

6. Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng sau khi chạy ứng dụng để xem giao diện trên các thiết bị khác nhau. Bạn đã tạo thành công một ứng dụng có giao diện phù hợp trên cả điện thoại và máy tính bảng với các kích thước và mật độ màn hình khác nhau.

Mẹo: Để có hướng dẫn chi tiết về cách sử dụng ConstraintLayout, hãy xem Sử dụng ConstraintLayout để thiết kế giao diện của bạn

Bước 2: Thay đổi bố cục thành LinearLayout

LinearLayout là một ViewGroup sắp xếp tập hợp các view theo hàng ngang hoặc hàng dọc. LinearLayout là một trong những layout phổ biến nhất vì nó đơn giản và nhanh chóng. Nó thường được sử dụng trong một ViewGroup khác để sắp xếp các phần tử giao diện theo chiều ngang hoặc dọc.

Một LinearLayout cần có các thuộc tính sau:

- layout_width
- layout_height
- orientation

Giá trị của layout_width và layout_height có thể là:

- **match_parent**: Mở rộng view để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao. Nếu **LinearLayout** là view gốc, nó sẽ mở rộng ra toàn bộ màn hình (phần tử cha).
- **wrap_content**: Thu nhỏ kích thước view sao cho vừa đủ với nội dung bên trong. Nếu không có nội dung, view sẽ trở nên vô hình.
- **Số dp cố định** (density-independent pixels - pixel độc lập với mật độ màn hình): Xác định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ: 16dp có nghĩa là 16 pixel độc lập với mật độ màn hình.

Giá trị của orientation có thể là:

- horizontal: Các view được sắp xếp theo chiều ngang (từ trái sang phải).
- vertical: Các view được sắp xếp theo chiều dọc (từ trên xuống dưới).

Trong bài này, bạn sẽ thay đổi ConstraintLayout của ứng dụng Hello Toast thành LinearLayout để thực hành sử dụng LinearLayout.

2.1 Thay đổi ViewGroup gốc thành LinearLayout

1. Mở ứng dụng Hello Toast từ tác vụ trước.
2. Mở tệp activity_main.xml (nếu chưa mở) và nhấp vào tab **Text** ở cuối ngăn chỉnh sửa để xem mã XML. Ở đầu mã XML, có dòng thẻ sau:

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Thay đổi thẻ <android.support.constraint.ConstraintLayout> thành <LinearLayout> sao cho mã trông như sau:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

4. Đảm bảo thẻ đóng ở cuối mã đã thay đổi thành </LinearLayout>. Nếu nó không tự động thay đổi, hãy chỉnh sửa thủ công.
5. Dưới dòng thẻ <LinearLayout>, thêm thuộc tính sau sau thuộc tính android:layout_height:

```
android:orientation="vertical">
```

Sau khi thực hiện các thay đổi này, một số thuộc tính XML của các phần tử khác sẽ bị gạch chân màu đỏ vì chúng được sử dụng với ConstraintLayout và không liên quan đến LinearLayout.

2.2 Thay đổi thuộc tính cho LinearLayout

Làm theo các bước sau để thay đổi thuộc tính của phần tử UI sao cho phù hợp với LinearLayout:

1. Mở ứng dụng Hello Toast từ nhiệm vụ trước.
2. Mở tệp activity_main.xml (nếu chưa mở) và nhấp vào tab Text.
3. Tìm phần tử button_toast và thay đổi thuộc tính sau:
4. Xóa các thuộc tính sau khỏi phần tử button_toast:
5. Tìm phần tử button_count và thay đổi thuộc tính sau:
6. Xóa các thuộc tính sau khỏi phần tử button_count:
7. Tìm phần tử show_count (TextView) và thay đổi các thuộc tính sau:
8. Xóa các thuộc tính sau khỏi phần tử show_count:
9. Nhấp vào tab Preview ở phía bên phải cửa sổ Android Studio (nếu chưa được chọn) để xem trước bố cục đã thay đổi.

```
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="?attr/colorPrimary"
    android:onClick="countUp"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"
    app:layout_constraintTop_toBottomOf="@+id/show_count" />

<Button
    android:id="@+id/button_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="?attr/colorPrimary"
    android:onClick="showToast"
    android:text="Toast"
    android:textColor="@android:color/white" />

<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:background="#FFF000"
    android:gravity="center_vertical"
    android:text="0"
    android:textAlignment="center"
    android:textColor="?attr/colorPrimary"
    android:textSize="160sp"
    android:textStyle="bold" />
```

2.3 Thay đổi vị trí các phần tử trong LinearLayout

LinearLayout sắp xếp các phần tử của nó theo một hàng ngang hoặc dọc. Bạn đã thêm thuộc tính android:orientation="vertical" cho LinearLayout, vì vậy các phần tử sẽ được xếp chồng lên nhau theo chiều dọc như trong hình trước.

Để thay đổi vị trí sao cho nút Count nằm ở dưới cùng, hãy làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ bài trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text**.
3. Chọn phần tử button_count (<Button>) cùng với tất cả các thuộc tính của nó (từ thẻ mở <Button> đến thẻ đóng />), sau đó chọn **Edit > Cut**.
4. Nhấp vào sau thẻ đóng /> của phần tử **TextView** nhưng trước thẻ đóng </LinearLayout>, sau đó chọn **Edit > Paste**.
5. *(Tùy chọn)* Để định dạng lại mã XML cho dễ đọc hơn, chọn **Code > Reformat Code** để tự động căn chỉnh khoảng cách và thụt dòng hợp lý.

Sau khi thực hiện, mã XML của các phần tử giao diện sẽ có dạng như sau:

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:background="?attr/colorPrimary"
    android:onClick="showToast"
    android:text="Toast"
    android:textColor="@android:color/white" />

<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:layout_marginTop="8dp"
    android:background="#FFF000"
    android:gravity="center_vertical"
    android:text="0"
    android:textAlignment="center"
    android:textColor="?attr/colorPrimary"
    android:textSize="160sp"
    android:textStyle="bold" />

<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="?attr/colorPrimary"
    android:onClick="countUp"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white" />
```

Bằng cách di chuyển nút button_count xuống dưới TextView, bố cục bây giờ gần giống như trước đó, với nút **Count** nằm ở phía dưới.

Bản xem trước của bố cục hiện trông như sau:

2.4 Thêm weight vào phần tử TextView

Xác định thuộc tính gravity và weight .Việc sử dụng thuộc tính gravity và weight giúp bạn kiểm soát tốt hơn cách sắp xếp các View và nội dung bên trong LinearLayout. android:gravity xác định cách căn chỉnh nội dung bên trong một View.Trong bài trước, bạn đã đặt thuộc tính này cho show_count TextView để căn giữa nội dung (chữ số "0") trong TextView.

```
android:gravity="center_vertical"
```

Thuộc tính android:layout_weight cho biết phần không gian dư thừa trong LinearLayout sẽ được phân bổ cho View như thế nào. Nếu chỉ một View có thuộc tính này, nó sẽ nhận toàn bộ không gian dư thừa. Nếu nhiều View có layout_weight, không gian sẽ được chia theo tỷ lệ. Ví dụ: Nếu mỗi Button có weight = 1 và TextView có weight = 2, tổng trọng số là 4. Khi đó, mỗi Button nhận ¼ không gian, còn TextView nhận ½ không gian.

Trên các thiết bị khác nhau, phần tử show_count TextView có thể chiếm một phần hoặc hầu hết không gian giữa các nút **Toast** và **Count**. Để đảm bảo TextView mở rộng và lấp đầy không gian có sẵn trên mọi thiết bị, hãy chỉ định thuộc tính android:gravity cho TextView.

Thực hiện các bước sau:

1. Mở ứng dụng Hello Toast từ bài trước.
2. Mở tệp activity_main.xml (nếu chưa mở) và nhấp vào tab Text.
3. Tìm phần tử show_count TextView và thêm thuộc tính sau:

Sau khi thực hiện, bản xem trước sẽ trông như hình minh họa sau.

Phần tử show_count TextView sẽ chiếm toàn bộ không gian giữa các nút. Bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ ở ngăn xem trước và chọn một thiết bị khác. Dù bạn chọn thiết bị nào để xem trước, phần tử show_count TextView vẫn sẽ lấp đầy toàn bộ không gian giữa các nút.

Bước 3: Thay đổi bố cục thành RelativeLayout

RelativeLayout là một nhóm bố cục trong đó mỗi View được định vị và căn chỉnh tương đối với các View khác trong cùng nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục bằng RelativeLayout.

3.1 Đổi từ LinearLayout sang RelativeLayout

Một cách dễ dàng để thay đổi LinearLayout thành RelativeLayou**t** là thêm các thuộc tính XML trong tab **Text**.

1. Mở tệp **activity_main.xml** và nhấp vào tab **Text** ở cuối khung chỉnh sửa để xem mã XML.
2. Thay đổi thẻ mở <LinearLayout> ở đầu tệp thành <**RelativeLayout**>, sao cho câu lệnh trông như sau:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Cuộn xuống để đảm bảo rằng thẻ đóng </LinearLayout> cũng đã được thay đổi thành </RelativeLayout>. Nếu chưa, hãy thay đổi nó thủ công.

3.2 Sắp xếp lại các View trong RelativeLayout

Cách dễ dàng để sắp xếp và định vị các View trong RelativeLayout là thêm các thuộc tính XML trong tab Text.

1. Nhấp vào tab **Preview** ở cạnh trình chỉnh sửa (nếu chưa được chọn) để xem bản xem trước bố cục, hiện tại trông giống như hình dưới đây.

Khi chuyển sang **RelativeLayout**, trình chỉnh sửa bố cục cũng đã thay đổi một số thuộc tính của các View. Ví dụ:

- Nút **Count** (button_count) đè lên nút Toast, vì vậy bạn không thể nhìn thấy nút **Toast** (button_toast).
- Phần trên của TextView (show_count) đè lên các nút.

2. Thêm thuộc tính android:layout_below vào nút button_count để đặt nút này ngay bên dưới show_count. Đây là một trong nhiều thuộc tính giúp định vị các View trong RelativeLayout, cho phép bạn đặt một View dựa trên vị trí của các View khác.
   android:layout_below="@+id/show_count"
3. Thêm thuộc tính android:layout_centerHorizontal vào cùng nút Button để căn giữa View theo chiều ngang trong thành phần cha của nó, trong trường hợp này là nhóm View RelativeLayout.
   android:layout_centerHorizontal="true"

Toàn bộ code XML của nút button_count :

```
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="?attr/colorPrimary"
    android:onClick="countUp"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"
    android:layout_below="@+id/show_count"
    android:layout_centerHorizontal="true"
    />
```

4. Thêm các thuộc tính sau vào show_count TextView:
   android:layout_below="@+id/button_toast"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true"
   Thuộc tính android:layout_alignParentLeft căn chỉnh view về phía bên trái của nhóm view cha RelativeLayout. Mặc dù thuộc tính này đủ để căn chỉnh view sang bên trái, nhưng bạn có thể muốn view căn sang bên phải nếu ứng dụng chạy trên thiết bị sử dụng ngôn ngữ từ phải sang trái. Do đó, thuộc tính android:layout_alignParentStart giúp cạnh "bắt đầu" của view này khớp với

cạnh bắt đầu của view cha. "Bắt đầu" là cạnh trái của màn hình nếu thiết bị sử dụng ngôn ngữ từ trái sang phải, hoặc là cạnh phải nếu thiết bị sử dụng ngôn ngữ từ phải sang trái.

5. Xóa thuộc tính android:layout_weight="1" khỏi show_count TextView, vì nó không liên quan đến RelativeLayout. Bản xem trước bố cục bây giờ sẽ trông giống như hình dưới đây.

Mẹo: RelativeLayout giúp bạn dễ dàng sắp xếp nhanh chóng các phần tử giao diện người dùng trong bố cục. Để tìm hiểu thêm về cách định vị các view trong RelativeLayout, hãy xem phần "Positioning Views" trong chủ đề "Relative Layout" của API Guide.

Tổng kết

Sử dụng trình chỉnh sửa bố cục để xem trước và tạo các biến thể:

- Để xem trước bố cục ứng dụng theo hướng ngang trong trình chỉnh sửa bố cục, nhấp vào nút **Orientation in Editor** trên thanh công cụ và chọn **Switch to Landscape**. Chọn **Switch to Portrait** để quay lại hướng dọc.
- Để tạo một biến thể của bố cục dành riêng cho hướng ngang, nhấp vào nút **Orientation in Editor** và chọn **Create Landscape Variation**. Một cửa sổ chỉnh sửa mới sẽ mở với tab **land/activity_main.xml** hiển thị bố cục dành cho chế độ ngang.
- Để xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thực hoặc trình giả lập, nhấp vào nút **Device in Editor** trên thanh công cụ, sau đó chọn một thiết bị.
- Để tạo một biến thể của bố cục dành riêng cho máy tính bảng (màn hình lớn hơn), nhấp vào nút **Orientation in Editor** và chọn **Create layout x-large Variation**. Một cửa sổ chỉnh sửa mới sẽ mở với tab **xlarge/activity_main.xml** hiển thị bố cục dành cho thiết bị có kích thước màn hình lớn.

Sử dụng ConstraintLayout:

- Để xóa tất cả ràng buộc trong bố cục có ConstraintLayout làm gốc, nhấp vào nút **Clear All Constraints** trên thanh công cụ.
- Bạn có thể căn chỉnh một phần tử giao diện người dùng có chứa văn bản (chẳng hạn như TextView hoặc Button) với một phần tử giao diện khác có chứa văn bản. Ràng buộc đường cơ sở (baseline constraint) giúp căn chỉnh phần tử sao cho các đường cơ sở của văn bản khớp nhau.
- Để tạo một ràng buộc đường cơ sở, di chuột qua phần tử giao diện người dùng cho đến khi nút baseline constraint xuất hiện bên dưới phần tử đó.
- Nút pack trên thanh công cụ cung cấp các tùy chọn để sắp xếp hoặc mở rộng các phần tử giao diện người dùng đã chọn. Bạn có thể sử dụng nó để sắp xếp các phần tử Button theo hàng ngang với khoảng cách đều nhau.

Sử dụng LinearLayout:

- LinearLayout là một ViewGroup sắp xếp các phần tử con theo hàng ngang hoặc hàng dọc.

- LinearLayout yêu cầu các thuộc tính layout_width, layout_height và orientation.
- match_parent cho layout_width hoặc layout_height: Mở rộng View để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao. Nếu LinearLayout là phần tử gốc, nó sẽ mở rộng theo kích thước màn hình.
- wrap_content cho layout_width hoặc layout_height: Thu nhỏ kích thước View sao cho vừa đủ với nội dung bên trong. Nếu không có nội dung, View sẽ trở nên vô hình.
- Sử dụng giá trị cố định dp (density-independent pixels) cho layout_width hoặc layout_height: Xác định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ: 16dp có nghĩa là 16 pixel độc lập với mật độ màn hình.
- Thuộc tính orientation của LinearLayout có thể là horizontal để sắp xếp các phần tử từ trái sang phải hoặc vertical để sắp xếp các phần tử từ trên xuống dưới.
- Thuộc tính android:gravity xác định cách căn chỉnh nội dung bên trong một View.
- Thuộc tính android:layout_weight chỉ định tỷ lệ phân bổ không gian trống trong LinearLayout. Nếu chỉ có một View có thuộc tính này, nó sẽ nhận toàn bộ không gian trống. Nếu có nhiều View, không gian được phân bổ theo tỷ lệ. Ví dụ, nếu hai Button có layout_weight = 1 và TextView có layout_weight = 2 (tổng là 4), mỗi Button nhận 1/4 không gian, còn TextView nhận 1/2 không gian.

Sử dụng RelativeLayout:

- RelativeLayout là một ViewGroup, trong đó mỗi View được định vị và căn chỉnh dựa trên các View khác trong cùng nhóm.
- Sử dụng android:layout_alignParentTop để căn View với mép trên của ViewGroup cha.
- Sử dụng android:layout_alignParentLeft để căn View với mép trái của ViewGroup cha.
- Sử dụng android:layout_alignParentStart để căn mép bắt đầu của View với mép bắt đầu của ViewGroup cha. Thuộc tính này hữu ích nếu bạn muốn ứng dụng hỗ trợ các thiết bị có ngôn ngữ hoặc cài đặt vùng khác nhau. Start là mép trái nếu ngôn ngữ được thiết lập theo hướng trái sang phải (LTR). Start là mép phải nếu ngôn ngữ được thiết lập theo hướng phải sang trái (RTL).

Bài về nhà

Thay đổi ứng dụng

Mở ứng dụng HelloToast.

1. Đổi tên dự án thành HelloConstraint, và tái cấu trúc dự án thành HelloConstraint. (Để biết hướng dẫn cách sao chép và tái cấu trúc một dự án, hãy xem phần Phụ lục: Tiện ích.)
2. Chỉnh sửa bố cục activity_main.xml để căn chỉnh các phần tử nút Toast và nút Count dọc theo phía bên trái của TextView show_count, hiển thị số "0". Tham khảo hình minh họa bên dưới để thiết kế bố cục.
3. Thêm một nút thứ ba có tên là Zero, hiển thị giữa nút Toast và nút Count.
4. Phân bổ đều các nút theo chiều dọc giữa phần trên và dưới của TextView show_count.
5. Thiết lập nút Zero ban đầu có nền màu xám.
6. Đảm bảo rằng bạn đã thêm nút Zero vào bố cục chế độ nằm ngang trong activity_main.xml (land), cũng như trên màn hình máy tính bảng trong activity_main.xml (xlarge).
7. Khi nhấn nút Zero, giá trị trong TextView show_count sẽ được đặt lại thành 0.
8. Cập nhật trình xử lý sự kiện (click handler) của nút Count để nó thay đổi màu nền của chính nó, tùy thuộc vào giá trị mới là số chẵn hay số lẻ. Gợi ý: Không sử dụng findViewById để tìm nút Count. Có cách nào khác bạn có thể sử dụng không? Bạn có thể sử dụng các hằng số trong lớp Color để đặt hai màu nền khác nhau.
9. Cập nhật trình xử lý sự kiện của nút Count để đặt màu nền của nút Zero thành màu khác ngoài màu xám, nhằm cho thấy rằng nó đã hoạt động. Gợi ý: Bạn có thể sử dụng findViewById trong trường hợp này.
10. Cập nhật trình xử lý sự kiện của nút Zero để đặt lại màu nền thành xám, để đảm bảo rằng nó luôn màu xám khi giá trị count = 0.

09:53
66%
HelloConstraint
Toast
Zero
Count
0

Trả lời các câu hỏi sau

Câu hỏi 1:

Hai thuộc tính ràng buộc bố cục nào trên nút Zero giúp căn chỉnh nó theo khoảng cách dọc bằng nhau giữa hai nút Button còn lại? (Chọn 2 câu trả lời.)

- **app:layout_constraintBottom_toTopOf="@+id/button_count"**
- android:layout_marginBottom="8dp"
- android:layout_marginStart="16dp"
- **app:layout_constraintTop_toBottomOf="@+id/button_toast"**
- android:layout_marginTop="8dp"

Câu hỏi 2:

Thuộc tính ràng buộc nào trên nút Zero giúp căn chỉnh nó theo chiều ngang với hai nút Button còn lại?

- **app:layout_constraintLeft_toLeftOf="parent"**
- app:layout_constraintBottom_toTopOf="@+id/button_count"
- android:layout_marginBottom="8dp"
- app:layout_constraintTop_toBottomOf="@+id/button_toast"

Câu hỏi 3:

Định dạng phương thức nào là đúng để sử dụng với thuộc tính android:onClick trong XML?

- public void callMethod()
- **public void callMethod(View view)**
- private void callMethod(View view)
- public boolean callMethod(View view)

Câu hỏi 4:

Trình xử lý nhấn vào nút Vount bắt đầu với định dạng phương thức :

public void countUp(View view)

Kỹ thuật nào hiệu quả hơn khi sử dụng trong trình xử lý này để thay đổi màu nền của nút Button?

- Sử dụng findViewById để tìm nút Count. Gán kết quả cho một biến View, sau đó sử dụng setBackgroundColor().
- **Sử dụng tham số view được truyền vào trình xử lý sự kiện với setBackgroundColor(): view.setBackgroundColor().**

1.4) Văn bản và các chế độ cuộn

Mở đầu

Lớp TextView là một lớp con của View, dùng để hiển thị văn bản trên màn hình. Bạn có thể kiểm soát cách văn bản xuất hiện bằng cách sử dụng các thuộc tính TextView trong tệp bố cục XML. Phần thực hành này hướng dẫn cách làm việc với nhiều phần tử TextView, bao gồm một phần tử mà người dùng có thể cuộn nội dung theo chiều dọc.

Nếu có nhiều thông tin hơn so với kích thước hiển thị của thiết bị, bạn có thể tạo một chế độ xem cuộn để người dùng có thể cuộn theo chiều dọc bằng cách vuốt lên hoặc xuống, hoặc cuộn theo chiều ngang bằng cách vuốt sang trái hoặc phải.

Thông thường, bạn sẽ sử dụng một chế độ xem cuộn cho các bài báo, tin tức hoặc bất kỳ đoạn văn bản dài nào không thể hiển thị đầy đủ trên màn hình. Bạn cũng có thể sử dụng chế độ xem cuộn để cho phép người dùng nhập nhiều dòng văn bản hoặc kết hợp các phần tử giao diện người dùng (chẳng hạn như trường nhập văn bản và nút) trong một chế độ xem cuộn.

Lớp ScrollView cung cấp bố cục cho chế độ xem cuộn. ScrollView là một lớp con của FrameLayout. Chỉ có một phần tử con bên trong nó—phần tử con này chứa toàn bộ nội dung cần cuộn. Phần tử con này có thể là một ViewGroup (chẳng hạn như LinearLayout) chứa các phần tử giao diện người dùng.

Các bố cục phức tạp có thể gặp vấn đề về hiệu suất khi chứa các phần tử con như hình ảnh. Một lựa chọn tốt cho một View trong ScrollView là LinearLayout được sắp xếp theo hướng dọc, hiển thị các phần tử mà người dùng có thể cuộn qua (chẳng hạn như các phần tử TextView).

Với ScrollView, tất cả các phần tử giao diện người dùng đều được lưu trong bộ nhớ và trong hệ thống phân cấp hiển thị ngay cả khi chúng không được hiển thị trên màn hình. Điều này giúp ScrollView trở thành lựa chọn lý tưởng để cuộn qua các trang văn bản dài một cách mượt mà vì nội dung văn bản đã có sẵn trong bộ nhớ. Tuy nhiên, ScrollView có thể tiêu tốn nhiều bộ nhớ, ảnh hưởng đến hiệu suất tổng thể của ứng dụng. Để hiển thị danh sách dài các mục mà người dùng có thể thêm, xóa hoặc chỉnh sửa, hãy cân nhắc sử dụng RecyclerView, được giới thiệu trong một bài học riêng.

Những gì bạn nên biết

Bạn có thể:

- Tạo ứng dụng Hello World với Android Studio.
- Chạy ứng dụng trên trình giả lập (emulator) hoặc thiết bị thực tế.
- Triển khai một TextView trong bố cục của ứng dụng.
- Tạo và sử dụng tài nguyên chuỗi (string resources).

Những gì bạn sẽ học

- Cách dùng mã XML để thêm nhiều phần tử TextView
- Cách dùng mã XML để định nghĩa một view có thể cuộn
- Cách hiển thị văn bản tự do với một số thẻ định dạng HTML.
- Cách tạo kiểu cho màu nền và màu chữ của TextView.
- Cách chèn liên kết web vào văn bản.

Những gì bạn sẽ làm

- Tạo ứng dụng ScrollingText
- Thay đổi ViewGroup ConstraintLayout thành RelativeLayout.
- Thêm hai phần tử TextView cho tiêu đề chính và phụ của bài viết.
- Sử dụng TextAppearance styles và màu sắc cho tiêu đề chính và phụ.
- Sử dụng thẻ HTML trong chuỗi văn bản để kiểm soát định dạng.
- Sử dụng thuộc tính lineSpacingExtra để tăng khoảng cách dòng, giúp văn bản dễ đọc hơn.
- Thêm ScrollView vào bố cục để cho phép cuộn trong phần tử TextView.
- Thêm thuộc tính autoLink để làm cho các URL trong văn bản có thể nhấp và hoạt động.

Tổng quan ứng dụng

Ứng dụng Scrolling Text minh họa cách sử dụng ScrollView, một ViewGroup chứa một TextView. Nó hiển thị một trang văn bản dài—trong trường hợp này là một bài đánh giá album nhạc—mà người dùng có thể cuộn dọc để đọc bằng cách vuốt lên và xuống. Một thanh cuộn xuất hiện ở lề phải.

Ứng dụng này cho thấy cách bạn có thể sử dụng văn bản được định dạng với các thẻ HTML tối thiểu để thiết lập văn bản in đậm hoặc in nghiêng, cũng như sử dụng ký tự xuống dòng để tách đoạn văn. Bạn cũng có thể chèn liên kết web có thể nhấp vào trong văn bản.

09:27 100%

Scrolling Text

Beatles Anthology Vol. 1

Behind That Locked Door: Beatles Rarities!

In a vault deep inside Abbey Road Studios in London — protected by an unmarked, triple-locked, police-alarmed door — are something like 400 hours of unreleased Beatles recordings, starting from June 2, 1962 and ending with the very last tracks recorded for the *Let It Be* album. The best of the best were released by Apple Records in the form of the 3-volume Anthology series. For more information, see the Beatles Time Capsule at www.rockument.com.

This volume starts with the first new Beatle song, “Free as a Bird” (based on a John Lennon demo, found only on *The Lost Lennon Tapes Vol. 28*, and covers the very earliest historical recordings, outtakes from the first albums, and live recordings from early concerts and BBC Radio sessions.

Highlights include:

Cry for a Shadow - Many a Beatle fanatic started down the outtake road, like I did, with a first listen to this song. Originally titled “Beatle Bop” and recorded in a single session that yielded four songs (the other three featured Tony Sheridan with the Beatles as a backing band), “Cry for a Shadow” is an instrumental written by Lennon and Harrison, which makes it unique to this day. John Lennon plays rhythm guitar, George Harrison plays lead guitar, Paul McCartney plays bass, and Pete Best plays drums. The sessions were produced by Bert Kaempfert in Hamburg, Germany, during the Beatles’ second visit from April

Trong hình trên, có các phần sau:

1. Một liên kết web có thể nhấp được nhúng trong văn bản tự do.
2. Thanh cuộn xuất hiện khi cuộn văn bản.

Bước 1 : Thêm và chỉnh sử phần tử TextView

Trong bài thực hành này, bạn sẽ tạo một dự án Android cho ứng dụng ScrollingText, thêm các TextView vào bố cục để hiển thị tiêu đề và phụ đề bài viết, đồng thời thay đổi phần tử TextView "Hello World" hiện có để hiển thị một bài viết dài.Hình minh họa dưới đây là sơ đồ bố cục của ứng dụng.

Bạn sẽ thực hiện tất cả các thay đổi này trong mã XML và trong tệp strings.xml. Bạn sẽ chỉnh sửa mã XML của bố cục trong tab Text, bằng cách nhấp vào tab Text thay vì tab Design để mở chế độ thiết kế. Một số thay đổi đối với các phần tử UI và thuộc tính sẽ dễ dàng thực hiện hơn khi chỉnh sửa mã nguồn XML trực tiếp trong tab Text.

1.1 Tạo dự án và phần tử TextView

Trong nhiệm vụ này, bạn sẽ tạo dự án và các phần tử TextView, đồng thời sử dụng các thuộc tính của TextView để định dạng văn bản và nền.

Mẹo: Để tìm hiểu thêm về các thuộc tính này, hãy tham khảo tài liệu TextView reference.

1. Trong Android Studio, tạo một dự án mới với các thông số sau:

2. Trong thư mục app > res > layout trong Project > Android pane, mở tệp activity_main.xml, sau đó nhấp vào tab Text để xem mã XML.
   Ở phần đầu hoặc gốc của cây View hierarchy, bạn sẽ thấy ConstraintLayout ViewGroup: **android.support.constraint.ConstraintLayout**
3. Thay đổi ViewGroup này thành RelativeLayout. Dòng mã thứ hai bây giờ sẽ trông giống như sau:

<RelativeLayout xmlns:android=http://schemas.android.com/apk/res/android

RelativeLayout cho phép bạn sắp xếp các phần tử giao diện người dùng (UI elements) theo vị trí tương đối với nhau hoặc so với RelativeLayout cha.

Phần tử TextView "Hello World" mặc định được tạo bởi mẫu Empty Layout có các thuộc tính ràng buộc (như app:layout_constraintBottom_toBottomOf="parent"). Đừng lo lắng—bạn sẽ xóa chúng trong các bước tiếp theo.

4. Xóa dòng mã XML sau, dòng này liên quan đến ConstraintLayout:
   xmlns:app="http://schemas.android.com/apk/res-auto"
   Bây giờ khối mã XML ở trên cùng trông như thế này:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

5. Thêm một phần tử TextView phía trên TextView "Hello World" bằng cách nhập: <TextView. Một khối TextView sẽ xuất hiện, kết thúc bằng /> và hiển thị các thuộc tính layout_width và layout_height, đây là các thuộc tính bắt buộc cho TextView.
6. Nhập các thuộc tính sau cho TextView. Khi bạn nhập từng thuộc tính và giá trị, các gợi ý sẽ xuất hiện để hoàn thành tên thuộc tính hoặc giá trị.

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/article_heading"
    android:background="@color/colorPrimary"
    android:textColor="@android:color/white"
    android:padding="10dp"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
    android:textStyle="bold"
    android:text="Article Title" />
```

7. Trích xuất tài nguyên chuỗi cho chuỗi được mã hóa cứng trong thuộc tính android:text của TextView với giá trị "Article Title" để tạo một mục cho nó trong strings.xml.
   Đặt con trỏ vào chuỗi được mã hóa cứng, nhấn Alt-Enter (hoặc Option-Enter trên Mac), và chọn Extract string resource. Đảm bảo rằng tùy chọn Create the resource in directories được chọn, sau đó chỉnh sửa tên tài nguyên cho giá trị chuỗi thành article_title.
   Tài nguyên chuỗi được mô tả chi tiết trong String Resources.
8. Trích xuất tài nguyên kích thước cho chuỗi được mã hóa cứng trong thuộc tính android:padding của TextView với giá trị "10dp" để tạo dimens.xml và thêm một mục vào đó. Đặt con trỏ vào chuỗi được mã hóa cứng, nhấn Alt-Enter (hoặc Option-Enter trên Mac), và chọn Extract dimension resource.

Đảm bảo rằng tùy chọn Create the resource in directories được chọn, sau đó chỉnh sửa tên tài nguyên thành padding_regular.

9. Thêm một phần tử TextView khác phía trên TextView có nội dung “Hello World” và phía dưới TextView mà bạn đã tạo trong các bước trước. Thêm các thuộc tính sau cho TextView:

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/article_subheading"
    android:layout_below="@id/article_heading"
    android:padding="@dimen/padding_regular"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault"
    android:text="Article Subtitle" />
```

Vì bạn đã trích xuất tài nguyên kích thước cho chuỗi "10dp" thành padding_regular trong TextView đã tạo trước đó, bạn có thể sử dụng @dimen/padding_regular cho thuộc tính android:padding trong TextView này.

10. Trích xuất tài nguyên chuỗi cho chuỗi được mã hóa cứng trong thuộc tính android:text của TextView với giá trị "Article Subtitle" thành article_subtitle.
11. Trong phần tử TextView có nội dung "Hello World", xóa các thuộc tính layout_constraint.
12. Thêm các thuộc tính TextView sau vào phần tử TextView có nội dung “Hello World” và thay đổi thuộc tính android:text :

```
<TextView
    android:id="@+id/article"
    android:layout_below="@id/article_subheading"
    android:lineSpacingExtra="5sp"
    android:padding="@dimen/padding_regular"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Article text" />
```

13. Trích xuất tài nguyên chuỗi cho chuỗi "Article text" thành article_text, và trích xuất tài nguyên kích thước cho "5sp" thành line_spacing.

14. Định dạng lại và căn chỉnh mã bằng cách chọn Code > Reformat Code. Đây là một thói quen tốt để giúp mã dễ đọc và dễ hiểu hơn cho bạn và những người khác.

1.2 Thêm nội dung cho bài viết

Trong một ứng dụng thực tế truy cập các bài báo từ tạp chí hoặc báo, các bài viết có thể đến từ một nguồn trực tuyến thông qua một content provider, hoặc có thể được lưu sẵn trong cơ sở dữ liệu trên thiết bị.

Trong bài thực hành này, bạn sẽ tạo bài viết dưới dạng một chuỗi dài trong tài nguyên strings.xml.

1. Trong thư mục app > res > values, mở tệp strings.xml.
2. Mở một tệp văn bản chứa nhiều nội dung hoặc mở tệp strings.xml của ứng dụng ScrollingText đã hoàn thành.
3. Nhập giá trị cho các chuỗi article_title và article_subtitle với tiêu đề và phụ đề do bạn tự tạo, hoặc sử dụng giá trị trong tệp strings.xml của ứng dụng ScrollingText đã hoàn thành. Chuỗi phải ở dạng một dòng, không chứa HTML tags hoặc nhiều dòng.
4. Nhập hoặc sao chép và dán văn bản vào chuỗi article_text.

Bạn có thể sử dụng nội dung từ tệp văn bản của mình hoặc văn bản trong tệp strings.xml của ứng dụng ScrollingText đã hoàn thành. Yêu cầu duy nhất: Nội dung phải đủ dài để không vừa trên màn hình.

Hãy ghi nhớ những điều sau (tham khảo hình minh họa bên dưới để có ví dụ cụ thể):

- Khi bạn nhập hoặc dán văn bản vào strings.xml, các dòng văn bản sẽ không tự động xuống dòng mà sẽ kéo dài sang bên phải. Đây là hành vi đúng, vì mỗi dòng bắt đầu từ lề trái thể hiện một đoạn văn bản đầy đủ. Nếu bạn muốn ngắt dòng, nhấn Enter để tạo kết thúc dòng cứng hoặc định dạng văn bản trước trong trình soạn thảo văn bản.
- Sử dụng \n để biểu thị kết thúc dòng và thêm một \n nữa để tạo dòng trống. Điều này giúp các đoạn văn không bị dính vào nhau.
- Nếu trong văn bản có dấu nháy đơn ('), bạn phải thoát nó bằng dấu gạch chéo ngược (\'). Tương tự, nếu có dấu nháy kép ("), bạn cũng phải thoát nó (\").

- Nếu có ký tự không thuộc ASCII, bạn cần thoát chúng để tránh lỗi. Xem phần Formatting and styling trong tài nguyên String Resources để biết thêm chi tiết.
- Sử dụng HTML tags cho định dạng văn bản: Bôi đậm: Dùng thẻ .... In nghiêng: Dùng thẻ .... Kết hợp cả hai: Dùng ... văn bản .... Lưu ý: Các thẻ HTML khác sẽ bị bỏ qua.
- Bao bọc toàn bộ nội dung bài viết trong thẻ <string name="article_text">...</string> trong tệp strings.xml.
- Thêm một liên kết web để kiểm tra, chẳng hạn như www.google.com. Không sử dụng thẻ HTML để tạo liên kết vì tất cả các thẻ khác ngoài và sẽ bị hiển thị dưới dạng văn bản thô.

1.3 Chạy ứng dụng

Chạy ứng dụng. Bài viết sẽ xuất hiện, nhưng người dùng không thể cuộn vì bạn chưa thêm ScrollView (bạn sẽ thực hiện điều này trong bước tiếp theo). Ngoài ra, khi nhấn vào liên kết web, không có phản hồi. Bạn cũng sẽ khắc phục điều này trong bước tiếp theo.

09:27 100%

Scrolling Text

Beatles Anthology Vol. 1

Behind That Locked Door: Beatles Rarities!

In a vault deep inside Abbey Road Studios in London — protected by an unmarked, triple-locked, police-alarmed door — are something like 400 hours of unreleased Beatles recordings, starting from June 2, 1962 and ending with the very last tracks recorded for the *Let It Be* album. The best of the best were released by Apple Records in the form of the 3-volume Anthology series. For more information, see the Beatles Time Capsule at www.rockument.com.

This volume starts with the first new Beatle song, “Free as a Bird” (based on a John Lennon demo, found only on *The Lost Lennon Tapes Vol. 28*, and covers the very earliest historical recordings, outtakes from the first albums, and live recordings from early concerts and BBC Radio sessions.

Highlights include:

Cry for a Shadow - Many a Beatle fanatic started down the outtake road, like I did, with a first listen to this song. Originally titled “Beatle Bop” and recorded in a single session that yielded four songs (the other three featured Tony Sheridan with the Beatles as a backing band), “Cry for a Shadow” is an instrumental written by Lennon and Harrison, which makes it unique to this day. John Lennon plays rhythm guitar, George Harrison plays lead guitar, Paul McCartney plays bass, and Pete Best plays drums. The sessions were produced by Bert Kaempfert in Hamburg, Germany, during the Beatles’ second visit from April

Mã của bước 1 :

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/article_heading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimary"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_title"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
        android:textColor="@android:color/white"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/article_subheading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/article_heading"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_subtitle"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault" />

    <TextView
        android:id="@+id/article"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/article_subheading"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_text" />

</RelativeLayout>
```

Bước 2 : Thêm ScrollView và kích hoạt liên kết web

Trong nhiệm vụ trước, bạn đã tạo ứng dụng **ScrollingText** với các phần tử TextView để hiển thị tiêu đề bài viết, phụ đề và nội dung bài viết dài. Bạn cũng đã thêm một liên kết web, nhưng liên kết này chưa hoạt động. Bạn sẽ thêm mã để làm cho nó có thể nhấn vào.

Ngoài ra, TextView không thể tự cuộn để hiển thị toàn bộ nội dung bài viết. Bạn sẽ thêm một ViewGroup mới có tên ScrollView vào bố cục XML để làm cho TextView có thể cuộn được.

2.1 Thêm thuộc tính autoLink để kích hoạt liên kết web

Thêm thuộc tính android:autoLink="web" vào TextView của bài viết. Mã XML của TextView này bây giờ sẽ như sau:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/article"
    android:autoLink="web"
    android:layout_below="@id/article_subheading"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
```

2.2 Thêm ScrollView vào bố cục

Để làm cho một View (chẳng hạn như TextView) có thể cuộn được, hãy đặt View bên trong một ScrollView.

1. Thêm một ScrollView giữa TextView của article_subheading và TextView của article_text. Khi nhập <ScrollView, Android Studio sẽ tự động thêm </ScrollView> ở cuối và hiển thị các gợi ý cho thuộc tính android:layout_width và android:layout_height.
2. Chọn wrap_content từ các gợi ý cho cả hai thuộc tính.

```
<TextView
    android:id="@+id/article_subheading"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_heading"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_subtitle"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault" />
<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_subheading">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/article"
    android:autoLink="web"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
</ScrollView>
```

3. Di chuyển thẻ đóng </ScrollView> xuống dưới để TextView của **article_text** nằm hoàn toàn bên trong ScrollView.
4. Xóa thuộc tính android:layout_below="@id/article_subheading" khỏi TextView của article_text và thêm nó vào ScrollView.

   android:layout_below="@id/article_subheading"

   Điều này đảm bảo ScrollView xuất hiện bên dưới TextView của article_subheading.
5. Chọn Code > Reformat Code để định dạng lại mã XML, giúp TextView của article_text được thụt lề bên trong ScrollView.
6. Nhấn vào tab Preview ở bên phải trình chỉnh sửa bố cục để xem trước giao diện.

09:53 100%

Scrolling Text

Beatles Anthology Vol. 1

Behind That Locked Door: Beatles Rarities!

In a vault deep inside Abbey Road Studios in London — protected by an unmarked, triple-locked, police-alarmed door — are something like 400 hours of unreleased Beatles recordings, starting from June 2, 1962 and ending with the very last tracks recorded for the *Let It Be* album. The best of the best were released by Apple Records in the form of the 3-volume Anthology series. For more information, see the Beatles Time Capsule at www.rockument.com.

This volume starts with the first new Beatle song, “Free as a Bird” (based on a John Lennon demo, found only on *The Lost Lennon Tapes Vol. 28*, and covers the very earliest historical recordings, outtakes from the first albums, and live recordings from early concerts and BBC Radio sessions.

Highlights include:

Cry for a Shadow - Many a Beatle fanatic started down the outtake road, like I did, with a first listen to this song. Originally titled “Beatle Bop” and recorded in a single session that yielded four songs (the other three featured Tony Sheridan with the Beatles as a backing band), “Cry for a Shadow” is an instrumental written by Lennon and Harrison, which makes it unique to this day. John Lennon plays rhythm guitar, George Harrison plays lead guitar, Paul McCartney plays bass, and Pete Best plays drums. The sessions were produced by Bert Kaempfert in Hamburg, Germany, during the Beatles’ second visit from April

2.3 Chạy ứng dụng

Kiểm tra cách văn bản cuộn:

1. Chạy ứng dụng trên thiết bị hoặc trình giả lập.

   Vuốt lên và xuống để cuộn bài viết. Thanh cuộn sẽ xuất hiện ở lề phải khi bạn cuộn.

   Nhấn vào liên kết web để mở trang web. Thuộc tính android:autoLink="web" sẽ tự động chuyển đổi URL có thể nhận diện (ví dụ: www.rockument.com) thành liên kết có thể nhấn.

2. Xoay thiết bị hoặc trình giả lập trong khi ứng dụng đang chạy. Quan sát cách ScrollView mở rộng để sử dụng toàn bộ màn hình nhưng vẫn cuộn đúng cách.
3. Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng.ScrollView sẽ mở rộng để tận dụng toàn bộ màn hình nhưng vẫn hoạt động bình thường khi cuộn.

Mã nguồn cho bước 2:

activity_main.xml × | strings.xml | colors.xml | values\themes.xm | night\themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/article_heading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimary"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_title"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
        android:textColor="@android:color/white"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/article_subheading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/article_heading"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_subtitle"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault" />

    <ScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/article_subheading"
        >

        <TextView
            android:id="@+id/article"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:autoLink="web"
            android:lineSpacingExtra="5sp"
            android:padding="10dp"
            android:text="In a vault deep inside Abbey Road Studios in London – p..." />
```

Bước 3: Cuộn nhiều phần tử

Như đã đề cập trước đó, ScrollView chỉ có thể chứa một phần tử con duy nhất (ví dụ: TextView của bài viết). Tuy nhiên, phần tử con này có thể là một ViewGroup khác chứa nhiều phần tử con, chẳng hạn như LinearLayout.

Để làm cho cả tiêu đề phụ (subheading) và bài viết cuộn cùng nhau, hãy bọc chúng trong một LinearLayout bên trong ScrollView. Khi đó, LinearLayout sẽ trở thành phần tử con duy nhất của ScrollView, và toàn bộ nội dung bên trong sẽ có thể cuộn được.

3.1 Thêm LinearLayout vào ScrollView

1. Mở tệp activity_main.xml của dự án ScrollingText và chọn tab Text để chỉnh sửa mã XML (nếu chưa được chọn).
2. Thêm một LinearLayout phía trên TextView của bài viết trong ScrollView. Khi nhập <LinearLayout>, Android Studio tự động thêm </LinearLayout>. Chọn match_parent cho android:layout_width và wrap_content cho android:layout_height.

```
<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_subheading">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    <TextView
        android:id="@+id/article"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:autoLink="web"
        android:lineSpacingExtra="5sp"
        android:padding="10dp"
```

match_parent giúp LinearLayout khớp với chiều rộng của ViewGroup cha. wrap_content giúp LinearLayout mở rộng đủ để chứa nội dung bên trong.

3. Di chuyển thẻ đóng </LinearLayout> xuống sau TextView của bài viết nhưng vẫn trước thẻ đóng </ScrollView>.
4. Thêm thuộc tính android:orientation="vertical" vào LinearLayout.
5. Chọn Code > Reformat Code để căn chỉnh mã XML đúng định dạng.

```
<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_subheading">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

    <TextView
        android:id="@+id/article"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:autoLink="web"
        android:lineSpacingExtra="5sp"
        android:padding="10dp"
        android:text="In a vault deep inside Abbey Road Studios in London — p..." />
    </LinearLayout>
</ScrollView>
```

3.2 Di chuyển các phần tử giao diện vào trong LinearLayout

Hiện tại, LinearLayout chỉ chứa một phần tử UI là article_text. Để cả tiêu đề phụ và nội dung bài viết cuộn cùng nhau, bạn cần di chuyển article_subheading vào LinearLayout.

1. Cắt (Edit > Cut) đoạn mã của article_subheading . Dán (Edit > Paste) đoạn mã lên trên article_text bên trong LinearLayout.
2. Xóa thuộc tính android:layout_below="@id/article_heading" khỏi article_subheading. Vì LinearLayout sắp xếp các phần tử theo chiều dọc, thuộc tính này không còn cần thiết.
3. Thay đổi thuộc tính android:layout_below của ScrollView từ **android:layout_below="@id/article_subheading"** sang **android:layout_below="@id/article_heading" .** Vì article_subheading đã nằm trong LinearLayout, ScrollView phải đặt bên dưới article_heading thay vì article_subheading.

Mã XML sau khi chỉnh sửa:

```
<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_heading">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:id="@+id/article_subheading"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="@dimen/padding_regular"
            android:text="@string/article_subtitle"
            android:textAppearance="@android:style/TextAppearance.DeviceDefault" />
    <TextView
        android:id="@+id/article"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:autoLink="web"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_text" />
    </LinearLayout>
</ScrollView>
```

4. Chạy ứng dụng
   Vuốt lên và xuống để cuộn bài viết, và lưu ý rằng tiêu đề phụ giờ đây cuộn cùng với bài viết, trong khi tiêu đề chính vẫn giữ nguyên vị trí.

Tổng kết

- Sử dụng ScrollView để cuộn một View con duy nhất (chẳng hạn như TextView). ScrollView chỉ có thể chứa một View hoặc ViewGroup.
- Sử dụng ViewGroup như LinearLayout làm phần tử con trong ScrollView để cuộn nhiều phần tử View. Bao bọc các phần tử trong LinearLayout.
- Hiển thị văn bản tự do trong TextView với các thẻ định dạng HTML cho chữ in đậm và in nghiêng.
- Sử dụng \n làm ký tự xuống dòng trong văn bản tự do để giữ các đoạn văn không dính liền nhau.

- Dùng thuộc tính android:autoLink="web" để làm cho các liên kết web trong văn bản có thể nhấp vào.

Bài tập về nhà

Thay đổi ứng dụng

Mở ứng dụng ScrollingText2 mà bạn đã tạo trong bài học Làm việc với các phần tử TextView.

1. Thay đổi tiêu đề phụ để nó bao bọc trong một cột bên trái có độ rộng 100 dp, như hình minh họa bên dưới.
2. Đặt văn bản của bài viết ở bên phải tiêu đề phụ, như hình minh họa bên dưới.

03:10 100%

Scrolling Text

Beatles Anthology Vol. 1

Behind That Locked Door: Beatles Rarities!

In a vault deep inside Abbey Road Studios in London — protected by an unmarked, triple-locked, police-alarmed door — are something like 400 hours of unreleased Beatles recordings, starting from June 2, 1962 and ending with the very last tracks recorded for the *Let It Be* album. The best of the best were released by Apple Records in the form of the 3-volume Anthology series. For more information, see the Beatles Time Capsule at www.rockument.com.

This volume starts with the first new Beatle song, “Free as a Bird” (based on a John Lennon demo, found only on *The Lost Lennon Tapes Vol. 28*, and covers the very earliest historical recordings, outtakes from the first albums, and live recordings from early concerts and BBC Radio sessions.

Highlights include:

Cry for a Shadow - Many a Beatle fanatic started down the outtake road, like I did, with a first listen to this song. Originally titled “Beatle Bop” and recorded in a single session that yielded four songs (the other three featured Tony Sheridan with the Beatles as a backing band), “Cry

Trả lời các câu hỏi

Câu hỏi 1

Bạn có thể sử dụng bao nhiêu View trong một ScrollView? Chọn một:

- Chỉ một View
- **Một View hoặc một ViewGroup**
- Bao nhiêu tùy ý

Câu hỏi 2

Thuộc tính XML nào bạn sử dụng trong LinearLayout để hiển thị các View cạnh nhau? Chọn một:

- **android:orientation="horizontal"**
- android:orientation="vertical"
- android:layout_width="wrap_content"

Câu hỏi 3

Thuộc tính XML nào bạn sử dụng để xác định độ rộng của LinearLayout bên trong ScrollView? Chọn một:

- android:layout_width="wrap_content"
- **android:layout_width="match_parent"**
- android:layout_width="200dp"

1.5) Tài nguyên có sẵn

Mở đầu

Những gì bạn cần biết trước

Bạn nên có khả năng:

- Hiểu quy trình làm việc cơ bản của Android Studio.
- Tạo một ứng dụng từ đầu bằng mẫu Empty Activity.
- Sử dụng trình chỉnh sửa bố cục (layout editor).

Những gì bạn sẽ học

- Nơi tìm thông tin và tài nguyên dành cho nhà phát triển.
- Cách thêm biểu tượng khởi chạy (launcher icon) cho ứng dụng của bạn.
- Cách tìm kiếm trợ giúp khi phát triển ứng dụng Android.

Những gì bạn sẽ làm

- Khám phá một số tài nguyên hữu ích dành cho nhà phát triển Android ở mọi cấp độ.
- Thêm biểu tượng khởi chạy (launcher icon) cho ứng dụng của bạn.

Tổng quan ứng dụng

Bạn sẽ thêm một biểu tượng khởi chạy (launcher icon) vào ứng dụng HelloToast mà bạn đã tạo trước đó hoặc vào một ứng dụng mới.

04:00
100%
Tìm kiếm
Máy tính
Samsung Notes
Game Launcher
Samsung Global Goals
HelloConstraint
My Application
The Hungry Developer
HelloWorld
Scrolling Text

Bước 1 : Thay đổi biểu tượng khởi chạy

Mỗi ứng dụng mới được tạo bằng Android Studio đều bắt đầu với một biểu tượng khởi chạy mặc định đại diện cho ứng dụng. Biểu tượng khởi chạy xuất hiện trong danh sách ứng dụng trên Google Play Store. Khi người dùng tìm kiếm trong Google Play Store, biểu tượng của ứng dụng sẽ hiển thị trong kết quả tìm kiếm.

Sau khi cài đặt ứng dụng, biểu tượng khởi chạy sẽ xuất hiện trên thiết bị ở nhiều vị trí khác nhau, bao gồm màn hình chính và màn hình tìm kiếm ứng dụng. Ví dụ, ứng dụng HelloToast xuất hiện trong màn hình Search Apps của trình giả lập với biểu tượng mặc định cho các dự án ứng dụng mới, như minh họa bên dưới.

03:20
100%
Tìm kiếm
Máy tính
Samsung Notes
Game Launcher
Samsung Global Goals
HelloConstraint
My Application
The Hungry Developer
HelloWorld
Scrolling Text

Thay đổi biểu tượng khởi chạy là một quy trình từng bước đơn giản, giúp bạn làm quen với các tính năng image asset của Android Studio. Trong nhiệm vụ này, bạn cũng sẽ tìm hiểu thêm về cách truy cập tài liệu chính thức của Android.

1.1 Khám phá tài liệu chính thức của Android

Bạn có thể tìm thấy tài liệu chính thức dành cho nhà phát triển Android tại **developer.android.com**.
Tài liệu này chứa rất nhiều thông tin và luôn được Google cập nhật.

1. Truy cập **developer.android.com/design/**
   Đây là phần nói về Material Design, một triết lý thiết kế giúp định hình giao diện và cách hoạt động của ứng dụng trên thiết bị di động. Điều hướng qua các liên kết để tìm hiểu thêm về Material Design. Ví dụ, truy cập phần Style để tìm hiểu về cách sử dụng màu sắc và các chủ đề khác.
2. Truy cập **developer.android.com/docs/**
   Tại đây, bạn có thể tìm thấy thông tin về API, tài liệu tham khảo, hướng dẫn, công cụ và các ví dụ mã nguồn.
3. Truy cập **developer.android.com/distribute/**
   Đây là nơi cung cấp thông tin về cách đưa ứng dụng lên Google Play, hệ thống phân phối kỹ thuật số của Google dành cho các ứng dụng được phát triển bằng Android SDK. Sử dụng Google Play Console để mở rộng số lượng người dùng và bắt đầu kiếm tiền từ ứng dụng của bạn.

1.2 Thêm một image asset cho biểu tượng khởi chạy

Để thêm một hình ảnh clip-art làm biểu tượng khởi chạy, hãy làm theo các bước sau:

1. Mở dự án HelloToast từ bài học trước về cách sử dụng layout editor, hoặc tạo một dự án ứng dụng mới.
2. Trong Project > Android pane, nhấp chuột phải (hoặc Control-click trên macOS) vào thư mục res, sau đó chọn New > Image Asset. Cửa sổ Configure Image Asset sẽ xuất hiện.

3. Trong trường Icon Type, chọn Launcher Icons (Adaptive & Legacy) nếu chưa được chọn.
4. Nhấp vào tab Foreground Layer, sau đó chọn Clip Art cho Asset Type.

5. Nhấp vào biểu tượng trong trường Clip Art. Các biểu tượng từ bộ Material Design sẽ xuất hiện.
6. Duyệt qua cửa sổ Select Icon, chọn một biểu tượng phù hợp (chẳng hạn như biểu tượng mood để gợi ý tâm trạng tốt), sau đó nhấp OK.

7. Nhấp vào tab Background Layer, chọn Color làm Asset Type, sau đó nhấp vào ô màu để chọn màu làm nền.
8. Nhấp vào tab Legacy và xem lại các cài đặt mặc định. Xác nhận rằng bạn muốn tạo các biểu tượng legacy, round và Google Play Store. Khi hoàn tất, nhấp Next.
9. Chạy ứng dụng.
   Android Studio tự động thêm các hình ảnh biểu tượng khởi chạy vào các thư mục mipmap cho các độ phân giải khác nhau. Kết quả là biểu tượng khởi chạy của ứng dụng sẽ thay đổi thành biểu tượng mới sau khi bạn chạy ứng dụng, như minh họa bên dưới.

04:00
100%
Tìm kiếm
Máy tính
Samsung Notes
Game Launcher
Samsung Global Goals
HelloConstraint
My Application
The Hungry Developer
HelloWorld
Scrolling Text

Bước 2: Sử dụng mẫu dự án

Android Studio cung cấp các mẫu dự án cho các thiết kế ứng dụng và hoạt động (activity) phổ biến, được khuyến nghị. Việc sử dụng các mẫu có sẵn giúp tiết kiệm thời gian và đảm bảo tuân theo các phương pháp thiết kế tốt nhất.

Mỗi mẫu đều bao gồm một activity khung sườn và giao diện người dùng. Bạn đã sử dụng mẫu Empty Activity trước đó. Mẫu Basic Activity có nhiều tính năng hơn và tích hợp các thành phần ứng dụng được khuyến nghị, chẳng hạn như menu tùy chọn hiển thị trên thanh ứng dụng (app bar).

2.1 Khám phá kiến trúc của Basic Activity

Mẫu Basic Activity là một mẫu linh hoạt do Android Studio cung cấp, giúp bạn khởi động quá trình phát triển ứng dụng một cách nhanh chóng.

1. Trong Android Studio, tạo một dự án mới với mẫu Basic Activity.
2. Xây dựng và chạy ứng dụng.
3. Xác định các phần được đánh dấu trong hình minh họa và bảng mô tả. Tìm các phần tương ứng trên màn hình thiết bị/emulator của bạn. Kiểm tra mã nguồn Java và các tệp XML được đề cập trong bảng.

Việc làm quen với mã nguồn Java và các tệp XML sẽ giúp bạn dễ dàng mở rộng và tùy chỉnh mẫu này theo nhu cầu của mình.

Kiến trúc của mẫu Basic Activity

#	Mô tả UI	Tham chiếu mã
1	Thanh trạng thái (Status Bar) Hệ thống Android cung cấp và kiểm soát thanh trạng thái.	Không hiển thị trong mã mẫu. Có thể truy cập từ activity của bạn. Ví dụ, bạn có thể ẩn thanh trạng thái nếu cần.
2	AppBarLayout > Toolbar	Trong activity_main.xml, tìm

	Thanh ứng dụng (còn gọi là thanh hành động) cung cấp cấu trúc trực quan, các thành phần giao diện tiêu chuẩn và điều hướng. Để đảm bảo tương thích ngược, AppBarLayout trong mẫu này nhúng một Toolbar với chức năng tương tự như ActionBar.	android.support.v7.widget.Toolbar bên trong android.support.design.widget.AppBarLayout. Thay đổi Toolbar để thay đổi giao diện của thành phần cha, App Bar. Để xem ví dụ, hãy tham khảo App Bar Tutorial.
3	Tên ứng dụng Được lấy từ tên gói (package name), nhưng bạn có thể đặt bất kỳ tên nào bạn muốn.	Trong AndroidManifest.xml: android:label="@string/app_name"
4	Nút tràn menu tùy chọn Chứa các mục menu cho activity, cũng như các tùy chọn toàn cục như Tìm kiếm (Search) và Cài đặt (Settings) của ứng dụng. Các mục menu của ứng dụng sẽ nằm trong menu này.	Trong MainActivity.java: onOptionsItemSelected() thực hiện các hành động khi một mục menu được chọn. Trong res > menu > menu_main.xml: Tệp tài nguyên xác định các mục menu cho menu tùy chọn.
5	Layout ViewGroup CoordinatorLayout là một ViewGroup giàu tính năng, cung cấp cơ chế để các phần tử giao diện người dùng (View/UI elements) tương tác với nhau. Giao diện người dùng của ứng dụng được đặt trong tệp content_main.xml, tệp này được bao gồm trong ViewGroup này.	Trong activity_main.xml: Không có View nào được xác định trực tiếp trong bố cục này. Thay vào đó, nó bao gồm một bố cục khác bằng lệnh include: @layout/content_main, nơi các View được xác định. Điều này giúp tách biệt các View hệ thống khỏi các View riêng của ứng dụng.
6	TextView Trong ví dụ, được sử dụng để hiển thị "Hello World".	Trong content_main.xml: Tất cả các UI elements của ứng dụng đều được định nghĩa trong tệp này.

	Thay thế nó bằng các UI elements phù hợp với ứng dụng của bạn.	
7	Nút hành động nổi (FAB)	Trong activity_main.xml, FAB được khai báo như một phần tử giao diện người dùng sử dụng biểu tượng clip-art. Trong MainActivity.java, một đoạn mã mẫu (stub) trong onCreate() được sử dụng để thiết lập onClick() listener cho FAB.

2.2 Tùy chỉnh ứng dụng được tạo từ mẫu

Thay đổi giao diện của ứng dụng được tạo từ mẫu Basic Activity. Ví dụ, bạn có thể thay đổi màu của thanh ứng dụng để khớp với thanh trạng thái (trên một số thiết bị, thanh trạng thái có thể có tông màu đậm hơn của cùng một màu chính). Bạn cũng có thể muốn xóa nút hành động nổi nếu không sử dụng nó.

1. Thay đổi màu của thanh ứng dụng (Toolbar) trong activity_main.xml bằng cách chỉnh thuộc tính android:background thành "?attr/colorPrimaryDark", giúp đặt màu của thanh ứng dụng thành màu chính tối hơn để khớp với thanh trạng thái: **android:background="?attr/colorPrimaryDark"**
2. Để xóa Floating Action Button (FAB), trước tiên hãy xóa đoạn mã khởi tạo trong onCreate() đặt onClick() listener cho nút này. Mở MainActivity và xóa đoạn mã sau:

```
binding.fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAnchorView(R.id.fab)
                .setAction( text: "Action", listener: null).show();
    }
});
```

3. Để xóa Floating Action Button (FAB) khỏi bố cục, hãy xóa đoạn mã XML sau từ activity_main.xml:

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    app:srcCompat="@android:drawable/ic_dialog_email" />
```

4. Thay đổi tên của ứng dụng hiển thị trên thanh ứng dụng bằng cách thay đổi app_name trong strings.xml thành:

```
<string name="app_name">New Application</string>
```

5. Chạy ứng dụng. Nút hành động nổi (Floating Action Button) không còn xuất hiện, tên đã thay đổi và màu nền của thanh ứng dụng (app bar) đã thay đổi.

05:03
100%
New Application
Next
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam in scelerisque sem. Mauris volutpat, dolor id interdum ullamcorper, risus dolor egestas lectus, sit amet mattis purus dui nec risus. Maecenas non sodales nisi, vel dictum dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Suspendisse blandit eleifend diam, vel rutrum tellus vulputate quis. Aliquam eget libero aliquet, imperdiet nisl a, ornare ex. Sed rhoncus est ut libero porta lobortis. Fusce in dictum tellus.
Suspendisse interdum ornare ante. Aliquam nec cursus lorem. Morbi id magna felis. Vivamus egestas, est a condimentum egestas, turpis nisl iaculis ipsum, in dictum tellus dolor sed neque. Morbi tellus erat, dapibus ut sem a, iaculis tincidunt dui. Interdum et malesuada fames ac ante ipsum primis in faucibus. Curabitur et eros porttitor, ultricies urna vitae, molestie nibh. Phasellus at commodo eros, non aliquet metus. Sed maximus nisl nec dolor bibendum, vel congue leo egestas.
Sed interdum tortor nibh, in sagittis risus mollis quis. Curabitur mi odio, condimentum sit amet auctor at, mollis non turpis. Nullam pretium libero vestibulum, finibus orci vel, molestie quam. Fusce blandit tincidunt nulla, quis sollicitudin libero facilisis et. Integer interdum nunc ligula, et fermentum metus hendrerit id. Vestibulum lectus felis, dictum at lacinia sit amet, tristique id quam. Cras eu consequat dui. Suspendisse sodales nunc ligula, in lobortis sem porta sed. Integer id ultrices magna, in luctus elit. Sed a pellentesque est.
Aenean nunc velit, lacinia sed dolor sed, ultrices viverra nulla. Etiam a venenatis nibh. Morbi laoreet, tortor sed facilisis varius, nibh orci rhoncus nulla, id elementum leo dui non lorem. Nam mollis ipsum quis auctor varius. Quisque elementum eu libero sed commodo. In eros

2.3 Khám phá cách thêm activity bằng cách sử dụng các mẫu

Trong các bài thực hành trước, bạn đã sử dụng các mẫu Empty Activity và Basic Activity. Trong các bài học sau, các mẫu bạn sử dụng sẽ khác nhau tùy theo nhiệm vụ.

Các mẫu activity này cũng có sẵn trong dự án của bạn, vì vậy bạn có thể thêm nhiều activity vào ứng dụng sau khi thiết lập dự án ban đầu. (Bạn sẽ tìm hiểu thêm về lớp Activity trong một chương khác.)

1. Tạo một dự án ứng dụng mới hoặc chọn một dự án hiện có.
2. Trong Project > Android pane, nhấp chuột phải vào thư mục java.
3. Chọn New > Activity > Gallery.
4. Thêm một Activity. Ví dụ, nhấp vào Navigation Drawer Activity để thêm một Activity có ngăn điều hướng vào ứng dụng.
5. Nhấp đúp vào các tệp bố cục (layout) của Activity để hiển thị chúng trong trình chỉnh sửa bố cục (layout editor).

Bước 3: Học từ mã nguồn mẫu

Android Studio và tài liệu Android cung cấp nhiều mã nguồn mẫu mà bạn có thể nghiên cứu, sao chép và tích hợp vào dự án của mình.

3.1 Mã nguồn mẫu Android

Bạn có thể khám phá hàng trăm mã nguồn mẫu trực tiếp trong Android Studio.

1. Trong Android Studio, chọn File > New > Import Sample.
2. Duyệt qua các mẫu có sẵn.
3. Chọn một mẫu và nhấp vào Next.
4. Chấp nhận các thiết lập mặc định và nhấp vào Finish.

Lưu ý: Các mẫu ở đây chỉ mang tính khởi điểm cho việc phát triển tiếp theo. Chúng tôi khuyến khích bạn thiết kế và xây dựng thêm ý tưởng của riêng mình vào đó.

3.2 Sử dụng SDK Manager để cài đặt tài liệu ngoại tuyến

Việc cài đặt Android Studio cũng bao gồm các thành phần thiết yếu của Android SDK (Software Development Kit). Tuy nhiên, bạn có thể cài đặt thêm thư viện và tài liệu thông qua SDK Manager.

1. Chọn Tools > Android > SDK Manager.
2. Trong cột bên trái, nhấp vào Android SDK.
3. Sao chép đường dẫn tại Android SDK Location ở đầu màn hình, vì bạn sẽ cần nó để tìm tài liệu trên máy tính.

4. Chuyển sang tab SDK Platforms. Tại đây, bạn có thể cài đặt các phiên bản hệ thống Android bổ sung.
5. Chuyển sang tab SDK Update Sites. Android Studio sẽ kiểm tra các trang web được liệt kê và chọn để cập nhật thường xuyên.
6. Chuyển sang tab SDK Tools. Bạn có thể cài đặt các công cụ SDK bổ sung chưa được cài mặc định, cũng như phiên bản ngoại tuyến của tài liệu dành cho nhà phát triển Android.
7. Tích chọn Documentation for Android SDK nếu nó chưa được cài đặt, sau đó nhấp vào Apply.
8. Khi quá trình cài đặt hoàn tất, nhấp vào Finish.
9. Điều hướng đến thư mục sdk mà bạn đã sao chép ở bước 3, sau đó mở thư mục docs.
10. Tìm tệp index.html và mở nó.

Bước 4: Nhiều nguồn tài nguyên hơn

- Kênh YouTube của Android Developer là một nguồn tuyệt vời cung cấp hướng dẫn và mẹo hữu ích.
- Nhóm Android đăng tin tức và mẹo trên blog chính thức của Android.

- Stack Overflow là một cộng đồng lập trình viên giúp đỡ lẫn nhau. Nếu gặp vấn đề, rất có thể ai đó đã đăng câu trả lời. Hãy thử đăng các câu hỏi như: “Làm thế nào để thiết lập và sử dụng ADB qua WiFi?” “Những lỗi rò rỉ bộ nhớ phổ biến nhất trong lập trình Android là gì?”
- Và cuối cùng, hãy tìm kiếm trên Google. Công cụ tìm kiếm của Google sẽ tổng hợp kết quả từ tất cả các nguồn này. Ví dụ: “Phiên bản hệ điều hành Android phổ biến nhất ở Ấn Độ là gì?”

4.1 Tìm kiếm trên Stack Overflow bằng thẻ (tags)

Truy cập Stack Overflow và nhập [android] vào ô tìm kiếm. Dấu ngoặc vuông [] cho biết bạn muốn tìm các bài viết có thẻ liên quan đến Android.

Bạn có thể kết hợp thẻ và từ khóa tìm kiếm để có kết quả cụ thể hơn. Hãy thử:

- [android] and [layout]
- [android] "hello world"

Để biết thêm về cách tìm kiếm trên Stack Overflow, hãy xem trung tâm trợ giúp của Stack Overflow.

Tổng kết

- Tài liệu chính thức dành cho nhà phát triển Android: developer.android.com
- Material Design là một triết lý thiết kế khái niệm, định hướng cách ứng dụng hiển thị và hoạt động trên thiết bị di động.
- Google Play Store là hệ thống phân phối kỹ thuật số của Google dành cho các ứng dụng được phát triển bằng Android SDK.
- Android Studio cung cấp các mẫu (templates) cho thiết kế ứng dụng và hoạt động (activity) phổ biến, giúp bạn có sẵn mã nguồn cho các trường hợp sử dụng thông thường.
- Khi tạo một dự án mới, bạn có thể chọn một mẫu hoạt động (activity template) cho màn hình đầu tiên của ứng dụng.
- Trong quá trình phát triển ứng dụng, bạn có thể tạo hoạt động (activities) và thành phần ứng dụng (app components) từ các mẫu dựng sẵn.
- Android Studio chứa nhiều mẫu mã nguồn (code samples) mà bạn có thể nghiên cứu, sao chép và tích hợp vào dự án của mình.

Bài tập về nhà

Tải một ứng dụng mẫu và khám phá tài nguyên

1. Tải một ứng dụng mẫu vào Android Studio.
2. Mở một trong các tệp Java activity của ứng dụng. Tìm một lớp (class), kiểu dữ liệu (type), hoặc phương thức (procedure) mà bạn chưa quen thuộc, sau đó tra cứu thông tin về nó trong tài liệu nhà phát triển Android.
3. Truy cập Stack Overflow và tìm kiếm các câu hỏi liên quan đến chủ đề đó.
4. Thay đổi biểu tượng khởi chạy (launcher icon). Sử dụng một biểu tượng có sẵn trong Image Asset của Android Studio.

Trả lời các câu hỏi:

Câu hỏi 1

Trong một dự án Android Studio, lệnh menu nào bạn có thể sử dụng để mở danh sách các ứng dụng mẫu?

Chọn một:

- File > Open
- **File > New > Import Sample**
- File > New > Import Module
- File > New > Import Project

Câu hỏi 2

Những nút nào được mẫu Basic Activity cung cấp trong giao diện người dùng? Chọn hai:

- Nút điều hướng (Navigation buttons)
- **Nút menu tùy chọn (Options menu overflow button)**
- **Nút hành động nổi (Floating action button)**
- Nút lớp Button với văn bản "Button"

Câu hỏi 3

Nguồn tài liệu nào là tài liệu chính thức dành cho nhà phát triển Android? Chọn một:

- stackoverflow.com
- officialandroid.blogspot.com
- **developer.android.com**
- github.com

Bài 2) Activities

2.1) Activity và Intent

Mở đầu

Một Activity đại diện cho một màn hình duy nhất trong ứng dụng của bạn, nơi người dùng có thể thực hiện một tác vụ tập trung, chẳng hạn như chụp ảnh, gửi email hoặc xem bản đồ. Một activity thường được hiển thị dưới dạng cửa sổ toàn màn hình cho người dùng.

Một ứng dụng thường bao gồm nhiều màn hình liên kết lỏng lẻo với nhau. Mỗi màn hình là một activity. Thông thường, một activity trong ứng dụng được chỉ định là activity chính (MainActivity.java), và nó sẽ được hiển thị khi ứng dụng khởi động. Activity chính có thể khởi chạy các activity khác để thực hiện các hành động khác nhau.

Mỗi khi một activity mới bắt đầu, activity trước đó sẽ bị dừng lại, nhưng hệ thống vẫn lưu activity đó vào một ngăn xếp (back stack). Khi một activity mới bắt đầu, nó sẽ được đẩy vào back stack và nhận sự tập trung từ người dùng. Back stack hoạt động theo nguyên tắc "vào sau, ra trước" (Last In, First Out - LIFO). Khi người dùng hoàn thành hoạt động hiện tại và nhấn nút Back, activity đó sẽ bị loại bỏ khỏi ngăn xếp và bị hủy, đồng thời activity trước đó sẽ được tiếp tục.

Một activity được khởi động hoặc kích hoạt thông qua một Intent. Intent là một thông điệp bất đồng bộ mà bạn có thể sử dụng trong activity của mình để yêu cầu một hành động từ một activity khác hoặc từ một thành phần khác trong ứng dụng. Bạn sử dụng

intent để khởi chạy một activity từ một activity khác và để truyền dữ liệu giữa các activity.

Có hai loại Intent:

- Explicit Intent (Intent tường minh): Là intent mà bạn biết rõ thành phần mục tiêu. Tức là, bạn đã biết tên lớp đầy đủ của activity cụ thể mà bạn muốn khởi chạy.
- Implicit Intent (Intent ẩn danh): Là intent mà bạn không biết rõ tên của thành phần mục tiêu, nhưng bạn có một hành động tổng quát cần thực hiện.

Trong bài thực hành này, bạn sẽ tạo explicit intents. Bạn sẽ tìm hiểu về implicit intents trong một bài thực hành sau.

Những gì bạn nên biết trước

Bạn nên có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.
- Sử dụng Layout Editor để tạo bố cục trong ConstraintLayout.
- Chỉnh sửa mã XML của bố cục.
- Thêm chức năng onClick cho một Button.

Những gì bạn sẽ học

- Cách tạo một Activity mới trong Android Studio.
- Cách xác định parent activity và child activity để điều hướng Up navigation.
- Cách khởi động một Activity bằng Explicit Intent.
- Cách truyền dữ liệu giữa các Activity bằng Explicit Intent.

Những gì bạn sẽ làm

- Tạo một ứng dụng Android mới với một Main Activity và một Second Activity.
- Truyền một chuỗi dữ liệu từ Main Activity sang Second Activity bằng Intent, và hiển thị dữ liệu đó trong Second Activity.

- Gửi một dữ liệu khác từ Second Activity trở lại Main Activity, cũng sử dụng Intent.

Tổng quan về ứng dụng

Trong chương này, bạn sẽ tạo và xây dựng một ứng dụng có tên Two Activities, đúng như tên gọi, ứng dụng này sẽ có hai Activity. Bạn sẽ phát triển ứng dụng qua ba giai đoạn.

Giai đoạn 1, tạo một ứng dụng trong đó Main Activity chứa một Button có tên Send. Khi người dùng nhấn nút Send, Main Activity sẽ sử dụng một Intent để khởi chạy Second Activity.

Giai đoạn 2, thêm một EditText vào Main Activity. Người dùng nhập một tin nhắn vào EditText, sau đó nhấn Send. Main Activity sẽ sử dụng một Intent để khởi chạy Second Activity và gửi tin nhắn của người dùng đến đó. Second Activity sẽ hiển thị tin nhắn đã nhận.

Giai đoạn 3 (Hoàn thiện ứng dụng) , thêm một EditText và một Reply Button vào Second Activity. Người dùng có thể nhập tin nhắn phản hồi và nhấn Reply. Tin nhắn phản hồi sẽ được hiển thị trên Main Activity. Lúc này, bạn sẽ sử dụng một Intent để gửi phản hồi từ Second Activity quay lại Main Activity.

Bước 1 : Tạo dự án TwoActivities project

Trong bước này, bạn sẽ thiết lập dự án ban đầu với một Main Activity, xác định bố cục giao diện, và tạo một phương thức khung (skeleton method) cho sự kiện onClick của nút bấm.

1.1 Tạo dự án TwoActivities project

1. Khởi động Android Studio và tạo một dự án Android Studio mới. Đặt tên ứng dụng là Two Activities. Chọn cài đặt Phone and Tablet giống như các bài thực hành trước. Thư mục dự án sẽ tự động được đặt tên là TwoActivities.Tên ứng dụng hiển thị trên App Bar sẽ là "Two Activities".
2. Chọn "Empty Activity" làm mẫu Activity. Nhấn Next.
3. Giữ nguyên tên Activity mặc định (MainActivity). Đảm bảo Generate Layout File và Backwards Compatibility (AppCompat) đều được chọn.
4. Nhấn "Finish" để hoàn tất việc tạo dự án.

1.2 Xác định bố cục cho Main Activity

1. Mở res > layout > activity_main.xml trong Project > Android pane. Trình chỉnh sửa bố cục (Layout Editor) sẽ xuất hiện.
2. Nhấp vào tab Design nếu nó chưa được chọn, và xóa TextView (dòng chữ "Hello World") trong Component Tree.
3. Khi Autoconnect đang bật (mặc định), kéo một Button từ Palette vào góc dưới bên phải của bố cục. Autoconnect sẽ tự động tạo các constraints cho Button.
4. Trong Attributes pane, thiết lập: ID: button_main , layout_width: wrap_content , layout_height: wrap_content , text: Send Bố cục bây giờ sẽ trông như thế này:

07:46
100%
Two Activities
Send

5. Nhấp vào tab Text để chỉnh sửa mã XML, thêm thuộc tính sau vào Button:

   android:onClick="launchSecondActivity"

   Giá trị thuộc tính này sẽ bị gạch chân màu đỏ vì phương thức launchSecondActivity() chưa được tạo. Bỏ qua lỗi này ngay bây giờ; bạn sẽ sửa nó trong bước tiếp theo.

6. Trích xuất chuỗi văn bản ("Send") thành một resource string, như đã hướng dẫn trong bài thực hành trước và đặt tên resource là button_main.

Sau khi hoàn thành, mã XML của Button sẽ trông giống như sau:

```
<Button
    android:id="@+id/button_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_main"
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
```

1.3 Xác định hành động của Button

1. Nhấp vào "launchSecondActivity" trong mã XML của activity_main.xml.

2. Nhấn Alt+Enter (Option+Enter trên Mac) và chọn Create 'launchSecondActivity(View)' in 'MainActivity'.

   Tệp MainActivity sẽ mở ra, và Android Studio sẽ tạo một phương thức khung cho trình xử lý launchSecondActivity().

3. Bên trong launchSecondActivity(), thêm một câu lệnh Log với nội dung "Button Clicked!".

```
public void launchSecondActivity(View view) {
    Log.d(LOG_TAG, msg: "Button clicked!");
}
```

LOG_TAG sẽ hiển thị màu đỏ. Bạn sẽ thêm định nghĩa cho biến đó trong một bước sau.

4. Ở đầu lớp MainActivity, thêm một hằng số cho biến LOG_TAG:
   private static final String LOG_TAG = MainActivity.class.getSimpleName();
   Hằng số này sử dụng chính tên của lớp làm tag.
5. Chạy ứng dụng của bạn. Khi bạn nhấn nút Send, bạn sẽ thấy thông báo "Button Clicked!" trong Logcat. Nếu có quá nhiều đầu ra trong Logcat, hãy nhập MainActivity vào ô tìm kiếm để chỉ hiển thị các dòng có tag đó.

Mã của MainActivity sẽ trông như sau:

```
public class MainActivity extends AppCompatActivity {
    1 usage
    private static final String LOG_TAG = MainActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }

    1 usage
    public void launchSecondActivity(View view) {
        Log.d(LOG_TAG,  msg: "Button clicked!");
    }
}
```

Bước 2 : Tạo và khởi chạy Avtivity thứ hai

Mỗi activity mới bạn thêm vào dự án sẽ có bố cục và tệp Java riêng, tách biệt với main activity. Chúng cũng có phần tử <activity> riêng trong tệp AndroidManifest.xml. Giống như main activity, các activity mới được tạo trong Android Studio cũng mở rộng từ lớp AppCompatActivity.

Mỗi activity trong ứng dụng của bạn chỉ được kết nối lỏng lẻo với các activity khác. Tuy nhiên, bạn có thể xác định một activity là parent của một activity khác trong AndroidManifest.xml. Mối quan hệ parent-child này cho phép Android thêm gợi ý điều hướng, chẳng hạn như mũi tên quay lại ở thanh tiêu đề của mỗi activity.

Một activity giao tiếp với các activity khác (trong cùng ứng dụng và trên các ứng dụng khác) bằng một intent. Intent có thể là:

- Explicit Intent: Khi bạn biết mục tiêu của intent, tức là bạn đã biết tên lớp đầy đủ của activity cụ thể đó.
- Implicit Intent: Khi bạn không có tên của thành phần đích, nhưng có một hành động chung cần thực hiện.

Trong nhiệm vụ này, bạn thêm một second activity vào ứng dụng, với bố cục riêng. Bạn chỉnh sửa AndroidManifest.xml để xác định main activity là parent của second activity. Sau đó, bạn chỉnh sửa phương thức launchSecondActivity() trong MainActivity để bao gồm một intent khởi chạy second activity khi bạn nhấp vào nút.

2.1 Tạo Activity thứ hai

1. Nhấp vào thư mục app trong dự án của bạn và chọn File > New > Activity > Empty Activity.
2. Đặt tên cho Activity mới là SecondActivity. Đảm bảo rằng Generate Layout File và Backwards Compatibility (AppCompat) được chọn. Tên layout sẽ tự động điền là activity_second. Không chọn tùy chọn Launcher Activity.
3. Nhấp vào Finish. Android Studio sẽ thêm cả layout mới (activity_second.xml) và tệp Java mới (SecondActivity.java) vào dự án của bạn cho Activity mới. Nó cũng cập nhật tệp AndroidManifest.xml để bao gồm Activity mới.

2.2 Chỉnh sửa tệp AndroidManifest.xml

1. Mở manifests > AndroidManifest.xml.
2. Tìm phần tử <activity> mà Android Studio đã tạo cho Second Activity:

   <activity android:name=".SecondActivity"></activity>

3. Thay thế toàn bộ phần tử <activity> bằng đoạn mã sau:

```
<activity android:name=".SecondActivity"
    android:label="Second Activity"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.example.twoactivities.MainActivity" />
</activity>
```

Thuộc tính label thêm tiêu đề của Activity vào thanh ứng dụng (app bar).

Thuộc tính parentActivityName xác định rằng MainActivity là parent của SecondActivity. Điều này được sử dụng cho Up navigation: Thanh ứng dụng của SecondActivity sẽ có mũi tên quay lại, giúp người dùng quay lại MainActivity.

Phần tử <meta-data> cung cấp thông tin bổ sung cho Activity dưới dạng cặp khóa-giá trị. Trong trường hợp này, nó làm cùng nhiệm vụ với android:parentActivityName, nhưng dành cho các phiên bản Android cũ (API < 16).

4. Trích xuất một string resource cho "Second Activity" trong đoạn mã trên và sử dụng tên tài nguyên là activity2_name.

2.3 Xác định bố cục cho Activity thứ hai

1. Mở activity_second.xml và nhấp vào tab Design nếu nó chưa được chọn.
2. Kéo một TextView từ Palette vào góc trên bên trái của bố cục, sau đó thêm ràng buộc (constraints) vào cạnh trên và cạnh trái của bố cục. Đặt các thuộc tính trong Attributes như sau:

Attribute	Value
id	text_header
Top margin	16
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	Message Received
textAppearance	AppCompat.Medium
textStyle	B (bold)

Giá trị của textAppearance là một thuộc tính chủ đề đặc biệt của Android, xác định các kiểu phông chữ cơ bản. Bạn sẽ tìm hiểu thêm về chủ đề (themes) trong bài học sau.

Bố cục bây giờ sẽ trông như thế này:

3. Nhấp vào tab Text để chỉnh sửa mã XML, và trích xuất chuỗi "Message Received" vào một tài nguyên có tên text_header.
4. Thêm thuộc tính android:layout_marginLeft="8dp" vào TextView để bổ sung cho thuộc tính layout_marginStart, giúp hỗ trợ các phiên bản Android cũ hơn.

Mã XML cho activity_second.xml sẽ như sau:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SecondActivity">

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header"
        android:textAppearance=
            "@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2.4 Thêm một Intent vào Activity chính

Trong nhiệm vụ này, bạn sẽ thêm một Intent tường minh (explicit Intent) vào Activity chính. Intent này được sử dụng để kích hoạt Activity thứ hai khi nút Send được nhấn.

1. Mở MainActivity.

2. Tạo một Intent mới trong phương thức launchSecondActivity().

   Constructor của Intent nhận hai tham số cho một Intent tường minh: một Context của ứng dụng và thành phần cụ thể sẽ nhận Intent đó. Ở đây, bạn nên sử dụng this làm Context và SecondActivity.class làm lớp cụ thể.

   Intent intent = new Intent(this, SecondActivity.class);

3. Gọi phương thức startActivity() với Intent mới làm đối số.

   startActivity(intent);

4. Chạy ứng dụng

   Khi bạn nhấn nút Send, MainActivity sẽ gửi Intent và hệ thống Android sẽ khởi chạy SecondActivity, hiển thị nó trên màn hình. Để quay lại MainActivity, hãy nhấn nút Up (mũi tên trái trên thanh ứng dụng) hoặc nút Back ở cuối màn hình.

Bước 3: Gửi dữ liệu từ Activity chính sang Activity thứ hai

Trong nhiệm vụ trước, bạn đã thêm một Intent tường minh vào MainActivity để khởi chạy SecondActivity. Bạn cũng có thể sử dụng Intent để gửi dữ liệu từ một Activity sang Activity khác khi khởi chạy nó.

Đối tượng Intent có thể truyền dữ liệu đến Activity đích theo hai cách: Trường dữ liệu (data field): Dữ liệu Intent có thể là một URI, chỉ định dữ liệu cụ thể cần xử lý.

Intent extras: Nếu thông tin bạn muốn truyền không phải là một URI hoặc bạn có nhiều hơn một phần thông tin cần gửi, bạn có thể đặt dữ liệu bổ sung vào extras.

Intent extras là các cặp key/value được lưu trữ trong một Bundle. Bundle là một tập hợp dữ liệu được lưu dưới dạng cặp key/value. Để truyền thông tin từ một Activity sang Activity khác, bạn đặt các key và value vào Bundle extras trong Activity gửi, sau đó lấy lại dữ liệu đó trong Activity nhận.

Trong nhiệm vụ này, bạn sẽ: Sửa đổi Intent trong MainActivity để đính kèm dữ liệu bổ sung (một chuỗi do người dùng nhập vào) trong Bundle extras. Chỉnh sửa SecondActivity để lấy dữ liệu từ Bundle extras của Intent và hiển thị nó trên màn hình.

3.1 Thêm một EditText cho MainActivity layout

1. Mở tệp activity_main.xml.
2. Kéo một phần tử Plain Text (EditText) từ bảng Palette vào phía dưới của bố cục. Thêm các ràng buộc (constraints): Bên trái của bố cục. Bên dưới của bố cục. Bên trái của nút Send. Thiết lập các thuộc tính trong bảng Attributes như sau:

```
<EditText
    android:id="@+id/editText_main"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginRight="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginBottom="16dp"
    android:ems="10"
    android:hint="@string/editText_main"
    android:inputType="textLongMessage"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button_main"
    app:layout_constraintStart_toStartOf="parent" />
```

Layout bây giờ sẽ trông như thế này :

3. Nhấn vào tab Text để chỉnh sửa mã XML , sau đó trích xuất chuỗi “Enter your Message Here” thành một tài nguyên có tên editText_main.

Mã XML cho bố cục sẽ trông giống như sau:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:text="Send"
        android:onClick="launchSecondActivity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

    <EditText
        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginRight="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginBottom="16dp"
        android:ems="10"
        android:hint="@string/editText_main"
        android:inputType="textLongMessage"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button_main"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3.2 Thêm một chuỗi vào Intent extras

Intent extras là các cặp key/value trong một Bundle. Bundle là một tập hợp dữ liệu được lưu trữ dưới dạng cặp key/value. Để truyền thông tin từ một Activity sang Activity khác, bạn đặt các key và value vào Intent extra Bundle.

1. Mở MainActivity.
2. Thêm một hằng số public ở đầu lớp để xác định key cho Intent extra:

public static final String EXTRA_MESSAGE =

"com.example.android.twoactivities.extra.MESSAGE";

3. Thêm một biến private ở đầu lớp để lưu trữ EditText.

private EditText mMessageEditText;

4. Trong phương thức onCreate(), sử dụng findViewById() để lấy tham chiếu đến EditText và gán nó vào biến private đó.

mMessageEditText = findViewById(R.id.editText_main);

5. Trong phương thức launchSecondActivity(), ngay bên dưới Intent mới, lấy văn bản từ EditText dưới dạng chuỗi:

String message = mMessageEditText.getText().toString();

6. Thêm chuỗi đó vào Intent dưới dạng extra, sử dụng hằng số EXTRA_MESSAGE làm key và chuỗi làm value:

intent.putExtra(EXTRA_MESSAGE, message);

Mã của MainActivity bây giờ sẽ trông như sau:

```
public class MainActivity extends AppCompatActivity {
    1 usage
    public static final String EXTRA_MESSAGE =
            "com.example.android.twoactivities.extra.MESSAGE";
    2 usages
    private EditText mMessageEditText;
    1 usage
    private static final String LOG_TAG = MainActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        mMessageEditText = findViewById(R.id.editText_main);
    }

    1 usage
    public void launchSecondActivity(View view) {
        Log.d(LOG_TAG,  msg: "Button clicked!");
        Intent intent = new Intent( packageContext: this, SecondActivity.class);
        String message = mMessageEditText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);

    }
}
```

3.3 Thêm một TextView vào SecondActivity để hiển thị thông điệp

1. Mở activity_second.xml.
2. Kéo một TextView khác vào bố cục, đặt bên dưới text_header và thêm các ràng buộc (constraints): Bên trái của bố cục. Bên dưới của text_header.
3. Thiết lập các thuộc tính cho TextView trong bảng Attributes như sau:

Attribute	Value
id	text_message
Top margin	8
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	(Delete any text in this field)
textAppearance	AppCompat.Medium

Bố cục mới trông giống như trong nhiệm vụ trước, vì TextView mới chưa chứa bất kỳ văn bản nào nên nó chưa xuất hiện trên màn hình.

Mã XML cho tệp activity_second.xml sẽ trông như sau:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SecondActivity">
    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header"
        android:textAppearance=
            "@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/text_header" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3.4 Chỉnh sửa SecondActivity để lấy dữ liệu từ extras và hiển thị thông điệp.

1. Mở SecondActivity và thêm mã vào phương thức onCreate().
2. Lấy Intent đã kích hoạt Activity này:

Intent intent = getIntent();

3. Lấy chuỗi chứa thông điệp từ Intent extras, sử dụng biến tĩnh MainActivity.EXTRA_MESSAGE làm key:

String message = getIntent().getStringExtra(MainActivity.EXTRA_MESSAGE);

4. Sử dụng findViewById() để lấy tham chiếu đến TextView hiển thị thông điệp từ bố cục:

TextView textView = findViewById(R.id.text_message);

5. Đặt văn bản của TextView thành chuỗi lấy từ Intent extra:

textView.setText(message);

6. Chạy ứng dụng. Khi bạn nhập một thông điệp trong MainActivity và nhấn Send, SecondActivity sẽ được khởi chạy và hiển thị thông điệp đó.

Phương thức onCreate() trong SecondActivity bây giờ sẽ trông như sau:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_second);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
    TextView textView = findViewById(R.id.text_message);
    textView.setText(message);

}
```

Bước 4: Trả dữ liệu về Activity chính

Bây giờ, khi ứng dụng của bạn đã có thể khởi chạy một Activity mới và gửi dữ liệu đến đó, bước cuối cùng là trả dữ liệu từ Activity thứ hai về Activity chính. Bạn cũng sẽ sử dụng Intent và Intent extras cho nhiệm vụ này.

4.1 Thêm một EditText và một Button vào bố cục của SecondActivity

1. Mở strings.xml và thêm các tài nguyên chuỗi cho văn bản của Button và gợi ý cho EditText mà bạn sẽ thêm vào SecondActivity:

```
<string name="button_second">Reply</string>
<string name="editText_second">Enter Your Reply Here</string>
```

2. Mở activity_main.xml và activity_second.xml.
3. Sao chép EditText và Button từ tệp bố cục activity_main.xml và dán chúng vào activity_second.xml.
4. Trong activity_second.xml, sửa đổi các giá trị thuộc tính cho Button như sau:

Old attribute value	New attribute value
`android:id="@+id/button_main"`	`android:id="@+id/button_second"`
`android:onClick= "launchSecondActivity"`	`android:onClick="returnReply"`
`android:text= "@string/button_main"`	`android:text= "@string/button_second"`

5. Trong activity_second.xml, sửa đổi các giá trị thuộc tính cho EditText như sau:

Old attribute value	New attribute value
`android:id="@+id/editText_main"`	`android:id="@+id/editText_second"`
`app:layout_constraintEnd_toStartOf="@ +id/button"`	`app:layout_constraintEnd_toStartOf="@ +id/button_second"`
`android:hint= "@string/editText_main"`	`android:hint= "@string/editText_second"`

6. Trong trình soạn thảo XML của Android Studio, nhấp vào returnReply, nhấn Alt+Enter (hoặc Option+Return trên Mac), và chọn Create 'returnReply(View)' in 'SecondActivity'.
   Android Studio sẽ tạo một phương thức khung cho trình xử lý returnReply(). Bạn sẽ triển khai phương thức này trong nhiệm vụ tiếp theo.

Bố cục mới cho activity_second.xml trông như sau:

Mã XML cho tệp activity_second.xml sẽ trông như sau:

```
<TextView
    android:id="@+id/text_header"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="16dp"
    android:text="Message Received"
    android:textAppearance=
        "@style/TextAppearance.AppCompat.Medium"
    android:textStyle="bold"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/text_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text_header" />
<Button
    android:id="@+id/button_second"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_second"
    android:onClick="returnReply"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent" />

<EditText
    android:id="@+id/editText_second"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginRight="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginBottom="16dp"
    android:ems="10"
    android:hint="Enter Your Reply Here"
    android:inputType="textLongMessage"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button_second"
    app:layout_constraintStart_toStartOf="parent" />
```

4.2 Tạo một Intent phản hồi trong SecondActivity

Dữ liệu phản hồi từ SecondActivity về MainActivity được gửi trong một Intent extra.

Bạn sẽ tạo Intent phản hồi và đặt dữ liệu vào đó theo cách tương tự như khi gửi Intent ban đầu.

1. Mở SecondActivity.
2. Ở đầu lớp, thêm một hằng số public để xác định key cho Intent extra.

public static final String EXTRA_REPLY =

"com.example.android.twoactivities.extra.REPLY";

3. Thêm một biến private ở đầu lớp để lưu trữ EditText.

private EditText mReply;

4. Trong phương thức onCreate(), trước đoạn mã Intent, sử dụng findViewById() để lấy tham chiếu đến EditText và gán nó vào biến private.

mReply = findViewById(R.id.editText_second);

5. Trong phương thức returnReply(), lấy văn bản từ EditText dưới dạng chuỗi.

String reply = mReply.getText().toString();

6. Trong phương thức returnReply(), tạo một Intent mới để phản hồi—không sử dụng lại đối tượng Intent nhận từ yêu cầu ban đầu.

Intent replyIntent = new Intent();

7. Thêm chuỗi phản hồi từ EditText vào Intent mới dưới dạng Intent extra. Ở đây, key là EXTRA_REPLY và value là nội dung phản hồi.

replyIntent.putExtra(EXTRA_REPLY, reply);

8. Đặt kết quả thành RESULT_OK để chỉ ra rằng phản hồi đã thành công. Lớp Activity định nghĩa các mã kết quả, bao gồm RESULT_OK và RESULT_CANCELED.

setResult(RESULT_OK,replyIntent);

9. Gọi finish() để đóng SecondActivity và quay lại MainActivity.

finish()

Mã cho SecondActivity bây giờ sẽ trông như sau:

```
public class SecondActivity extends AppCompatActivity {
    1 usage
    public static final String EXTRA_REPLY =
            "com.example.android.twoactivities.extra.REPLY";
    2 usages
    private EditText mReply;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_second);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
        TextView textView = findViewById(R.id.text_message);
        textView.setText(message);
        mReply = findViewById(R.id.editText_second);
    }

    1 usage
    public void returnReply(View view) {
        String reply = mReply.getText().toString();
        Intent replyIntent = new Intent();
        replyIntent.putExtra(EXTRA_REPLY, reply);
        setResult(RESULT_OK,replyIntent);
        finish();
    }
}
```

4.3 Thêm các phần tử TextView để hiển thị phản hồi

MainActivity cần một cách để hiển thị phản hồi mà SecondActivity gửi về. Trong nhiệm vụ này, bạn sẽ thêm các phần tử TextView vào bố cục activity_main.xml để hiển thị phản hồi trong MainActivity.

Để thực hiện nhanh hơn, bạn có thể sao chép các phần tử TextView đã sử dụng trong SecondActivity.

1. Mở strings.xml và thêm một tài nguyên chuỗi cho tiêu đề phản hồi.

<string name="text_header_reply">Reply Received</string>

2. Mở activity_main.xml và activity_second.xml.
3. Sao chép hai phần tử TextView từ tệp bố cục activity_second.xml và dán chúng vào activity_main.xml, phía trên Button.
4. Trong activity_main.xml, chỉnh sửa các giá trị thuộc tính cho TextView đầu tiên như sau:

Old attribute value	New attribute value
`android:id="@+id/text_header"`	`android:id="@+id/text_header_reply"`
`android:text= "@string/text_header"`	`android:text= "@string/text_header_reply"`

5. Trong activity_main.xml, chỉnh sửa các giá trị thuộc tính cho TextView thứ hai như sau:

Old attribute value	New attribute value
`android:id="@+id/text_message"`	`android:id= "@+id/text_message_reply"`
`app:layout_constraintTop_toBottomOf="@+id/text_header"`	`app:layout_constraintTop_toBottomOf= "@+id/text_header_reply"`

6. Thêm thuộc tính android:visibility="invisible" cho mỗi TextView để chúng ban đầu không hiển thị.
   (Việc hiển thị các phần tử này trên màn hình mà không có nội dung có thể gây nhầm lẫn cho người dùng.)

Bạn sẽ làm cho các TextView này hiển thị sau khi dữ liệu phản hồi được gửi về từ SecondActivity.

Bố cục activity_main.xml trông giống như trong nhiệm vụ trước—mặc dù bạn đã thêm hai TextView vào bố cục. Vì bạn đặt chúng ẩn, chúng sẽ không xuất hiện trên màn hình.

Mã XML cho tệp activity_main.xml sẽ trông như sau:

```
<TextView
    android:id="@+id/text_header_reply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:text="@string/text_header_reply"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    android:textStyle="bold"
    android:visibility="invisible"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<TextView
    android:id="@+id/text_message_reply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text_header_reply" />
<Button
    android:id="@+id/button2"android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_main"
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
<EditText
    android:id="@+id/editText_main"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:ems="10"
    android:hint="@string/editText_main"
    android:inputType="textLongMessage"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button2"
    app:layout_constraintStart_toStartOf="parent" />
```

4.4 Lấy phản hồi từ Intent extra và hiển thị nó

Khi bạn sử dụng một Intent tường minh để khởi chạy một Activity khác, bạn có thể không mong đợi nhận lại dữ liệu—bạn chỉ đơn giản kích hoạt Activity đó. Trong trường hợp này, bạn sử dụng startActivity() để khởi chạy Activity mới, như bạn đã làm trước đó. Tuy nhiên, nếu bạn muốn nhận dữ liệu phản hồi từ Activity được kích hoạt, bạn cần khởi chạy nó bằng startActivityForResult().

Trong nhiệm vụ này, bạn chỉnh sửa ứng dụng để khởi chạy SecondActivity với kỳ vọng nhận kết quả, trích xuất dữ liệu phản hồi từ Intent, và hiển thị dữ liệu đó trong các phần tử TextView mà bạn đã tạo trong nhiệm vụ trước.

1. Mở MainActivity.
2. Thêm một hằng số public ở đầu lớp để xác định key cho loại phản hồi cụ thể mà bạn quan tâm.

```
public static final int TEXT_REQUEST = 1;
```

3. Thêm hai biến private để lưu trữ reply header và reply TextView.

```
private TextView mReplyHeadTextView;

private TextView mReplyTextView;
```

4. Trong phương thức onCreate(), sử dụng findViewById() để lấy tham chiếu từ bố cục đến reply header và reply TextView, sau đó gán các tham chiếu đó cho các biến private.

```
mReplyHeadTextView = findViewById(R.id.text_header_reply);

mReplyTextView = findViewById(R.id.text_message_reply);
```

Phương thức onCreate() đầy đủ bây giờ sẽ như sau:

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
    mMessageEditText = findViewById(R.id.editText_main);
    mReplyHeadTextView = findViewById(R.id.text_header_reply);
    mReplyTextView = findViewById(R.id.text_message_reply);

}
```

5. Trong phương thức launchSecondActivity(), thay đổi lời gọi startActivity() thành startActivityForResult(), và thêm TEXT_REQUEST làm đối số.

startActivityForResult(intent, TEXT_REQUEST);

6. Ghi đè phương thức onActivityResult() với chữ ký sau:

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) { }

Các đối số của onActivityResult() chứa tất cả thông tin cần thiết để xử lý dữ liệu phản hồi: requestCode: Mã yêu cầu mà bạn đặt khi khởi chạy Activity bằng startActivityForResult(). resultCode: Mã kết quả do Activity đã khởi chạy đặt (thường là RESULT_OK hoặc RESULT_CANCELED). Intent data: Chứa dữ liệu được trả về từ Activity đã khởi chạy.

7. Bên trong onActivityResult(), gọi super.onActivityResult().

super.onActivityResult(requestCode, resultCode, data)

8. Thêm mã để kiểm tra TEXT_REQUEST nhằm đảm bảo bạn xử lý đúng kết quả của Intent, và kiểm tra RESULT_OK để đảm bảo yêu cầu thành công.

Lớp Activity định nghĩa các mã kết quả sau: RESULT_OK (yêu cầu thành công). RESULT_CANCELED (người dùng đã hủy thao tác). RESULT_FIRST_USER (dành cho việc định nghĩa mã kết quả riêng).

9. Bên trong khối if, lấy Intent extra từ Intent phản hồi (data). Ở đây, key cho extra là EXTRA_REPLY từ SecondActivity.

String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);

10. Đặt thuộc tính visibility của reply header thành true.

mReplyHeadTextView.setVisibility(View.VISIBLE);

11. Cập nhật nội dung của reply TextView với nội dung phản hồi, và đặt visibility của nó thành true.

mReplyTextView.setText(reply);

mReplyTextView.setVisibility(View.VISIBLE);

Phương thức onActivityResult() đầy đủ bây giờ sẽ như sau:

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == TEXT_REQUEST) {
        if (resultCode == RESULT_OK) {
            String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
            mReplyHeadTextView.setVisibility(View.VISIBLE);
            mReplyTextView.setText(reply);
            mReplyTextView.setVisibility(View.VISIBLE);
        }
    }
}
```

12. Chạy ứng dụng.

Kết quả: Khi bạn gửi một tin nhắn đến SecondActivity và nhận phản hồi, MainActivity sẽ cập nhật để hiển thị phản hồi đó.

04:04
100%
Two Activities
Reply Received
this is the reply
Enter Your Message Here
Send

Tổng kết

Tổng quan:

- Activity là một thành phần của ứng dụng cung cấp một màn hình tập trung vào một tác vụ duy nhất của người dùng.
- Mỗi Activity có một tệp giao diện người dùng (layout file) riêng.
- Bạn có thể thiết lập mối quan hệ cha/con giữa các Activity để hỗ trợ điều hướng Up Navigation trong ứng dụng.
- Một View có thể được hiển thị hoặc ẩn bằng thuộc tính android:visibility.

Cách triển khai một Activity:

- Chọn File > New > Activity để bắt đầu từ một mẫu có sẵn và tự động thực hiện các bước cần thiết.
- Nếu không bắt đầu từ mẫu có sẵn, bạn cần:
    1. Tạo một lớp Activity bằng Java/Kotlin.
    2. Xây dựng giao diện người dùng (UI) cho Activity trong tệp XML layout liên kết.
    3. Khai báo Activity mới trong AndroidManifest.xml.

Intent:

- Intent cho phép bạn yêu cầu một hành động từ một thành phần khác trong ứng dụng, ví dụ: khởi chạy một Activity từ một Activity khác.
- Intent có thể là tường minh (explicit) hoặc ngầm định (implicit).
    - Explicit Intent: Chỉ định rõ thành phần đích sẽ nhận dữ liệu.
    - Implicit Intent: Chỉ định chức năng mong muốn nhưng không chỉ rõ thành phần đích.
- Intent có thể chứa dữ liệu cần xử lý (URI) hoặc thông tin bổ sung dưới dạng Intent extras.
- Intent extras là cặp key/value trong một Bundle, được gửi kèm theo Intent.

Bài tập về nhà

Xây dựng và chạy ứng dụng

Mở ứng dụng HelloToast mà bạn đã tạo trong bài thực hành trước.

1. Chỉnh sửa nút Toast để khi nhấn vào, nó sẽ khởi chạy một Activity mới hiển thị từ "Hello!" và số đếm hiện tại như hình minh họa dưới đây.
2. Thay đổi văn bản trên nút Toast thành "Say Hello".

04:46
100%
Hello Toast
Say Hello
5
Count

04:47
100%
Hello Toast
Hello!
5

Trả lời các câu hỏi sau

Câu 1

Khi bạn thêm một Activity thứ hai vào ứng dụng bằng cách chọn File > New > Activity và một mẫu Activity, những thay đổi nào sẽ được thực hiện? Chọn một đáp án:

● Activity thứ hai được thêm dưới dạng một lớp Java. Bạn vẫn cần thêm tệp XML layout.

● Tệp XML layout của Activity thứ hai được tạo và một lớp Java được thêm vào. Bạn vẫn cần xác định chữ ký của lớp.

● Activity thứ hai được thêm dưới dạng một lớp Java, tệp XML layout được tạo và tệp AndroidManifest.xml được thay đổi để khai báo Activity thứ hai.

● Tệp XML layout của Activity thứ hai được tạo và tệp AndroidManifest.xml được thay đổi để khai báo Activity thứ hai.

Câu 2

Điều gì xảy ra nếu bạn xóa android:parentActivityName và phần tử <meta-data> khỏi khai báo Activity thứ hai trong tệp AndroidManifest.xml? Chọn một đáp án:

● Activity thứ hai không còn xuất hiện khi bạn cố gắng khởi chạy nó bằng một explicit Intent.

● Tệp XML layout của Activity thứ hai bị xóa.

● Nút Back không còn hoạt động trong Activity thứ hai để đưa người dùng quay lại Activity chính.

● Nút Up trên thanh ứng dụng không còn xuất hiện trong Activity thứ hai để đưa người dùng quay lại Activity cha.

Câu 3

Bạn sử dụng phương thức khởi tạo nào để tạo một explicit Intent mới? Chọn một đáp án:

● new Intent()

● new Intent(Context context, Class<?> class)

● new Intent(String action, Uri uri)

● new Intent(String action)

Câu 4

Trong bài tập ứng dụng HelloToast, bạn thêm giá trị hiện tại của bộ đếm vào Intent như thế nào? Chọn một đáp án:

● Dưới dạng dữ liệu của Intent (Intent data)

● Dưới dạng Intent TEXT_REQUEST

● Dưới dạng một hành động của Intent (Intent action)

● **Dưới dạng một Intent extra**

Câu 5

Trong bài tập ứng dụng HelloToast, làm thế nào để hiển thị giá trị hiện tại của bộ đếm trong Activity "Hello" thứ hai? Chọn một đáp án:

● Nhận Intent đã được sử dụng để khởi chạy Activity.

● Lấy giá trị bộ đếm hiện tại từ Intent.

● Cập nhật TextView để hiển thị giá trị bộ đếm.

● **Tất cả các đáp án trên.**

2.2) Vòng đời của Activity và trạng thái

Mở đầu

Trong bài thực hành này, bạn sẽ tìm hiểu thêm về vòng đời của Activity. Vòng đời là tập hợp các trạng thái mà một Activity có thể trải qua trong suốt thời gian tồn tại của nó, từ khi được tạo ra cho đến khi bị hủy và hệ thống thu hồi tài nguyên. Khi người dùng điều hướng giữa các Activity trong ứng dụng của bạn (cũng như vào và ra khỏi ứng dụng), các Activity sẽ chuyển đổi giữa các trạng thái khác nhau trong vòng đời của chúng.

Mỗi giai đoạn trong vòng đời của một Activity có một phương thức gọi lại (callback) tương ứng: onCreate(), onStart(), onPause(), v.v. Khi một Activity thay đổi trạng thái, phương thức gọi lại liên quan sẽ được kích hoạt. Bạn đã từng thấy một trong những phương thức này: onCreate(). Bằng cách ghi đè (override) bất kỳ phương thức gọi lại nào trong lớp Activity của mình, bạn có thể thay đổi hành vi mặc định của Activity để phản hồi các hành động của người dùng hoặc hệ thống.

Trạng thái của Activity cũng có thể thay đổi khi có sự thay đổi về cấu hình thiết bị, chẳng hạn như khi người dùng xoay thiết bị từ chế độ dọc sang ngang. Khi xảy ra những thay đổi cấu hình này, Activity sẽ bị hủy và khởi tạo lại ở trạng thái mặc định, có thể khiến người dùng mất thông tin mà họ đã nhập. Để tránh gây nhầm lẫn cho người dùng, điều quan trọng là bạn phải phát triển ứng dụng sao cho có thể ngăn chặn việc mất dữ liệu ngoài ý muốn. Trong phần sau của bài thực hành này, bạn sẽ thử nghiệm với các thay đổi cấu hình và tìm hiểu cách duy trì trạng thái của Activity khi thiết bị thay đổi cấu hình hoặc khi có các sự kiện liên quan đến vòng đời của Activity.

Trong bài thực hành này, bạn sẽ thêm các câu lệnh ghi log vào ứng dụng TwoActivities và quan sát các thay đổi trong vòng đời của Activity khi sử dụng ứng dụng. Sau đó, bạn sẽ bắt đầu làm việc với các thay đổi này và khám phá cách xử lý đầu vào của người dùng trong những điều kiện này.

Những gì bạn cần biết trước

Bạn nên có khả năng:

- Tạo và chạy một dự án ứng dụng trong Android Studio.
- Thêm các câu lệnh log vào ứng dụng và xem chúng trong Logcat.
- Hiểu và làm việc với Activity và Intent, đồng thời cảm thấy thoải mái khi tương tác với chúng.

Những gì bạn sẽ học

- Cách hoạt động của vòng đời Activity.
- Khi nào một Activity bắt đầu, tạm dừng, dừng và bị hủy.
- Các phương thức gọi lại (callback) trong vòng đời của Activity.
- Ảnh hưởng của các hành động (chẳng hạn như thay đổi cấu hình) dẫn đến sự kiện vòng đời của Activity.
- Cách duy trì trạng thái Activity khi xảy ra các sự kiện vòng đời.

Những gì bạn sẽ làm

- Thêm mã vào ứng dụng TwoActivities từ bài thực hành trước để triển khai các phương thức callback của vòng đời Activity, bao gồm các câu lệnh ghi log.
- Quan sát sự thay đổi trạng thái khi ứng dụng chạy và khi bạn tương tác với từng Activity trong ứng dụng.
- Chỉnh sửa ứng dụng để giữ lại trạng thái của Activity khi nó bị hủy và khởi tạo lại do hành vi của người dùng hoặc thay đổi cấu hình thiết bị.

Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ tiếp tục phát triển ứng dụng TwoActivities. Ứng dụng sẽ trông và hoạt động tương tự như trong bài hướng dẫn trước. Nó bao gồm hai Activity và cung cấp cho người dùng khả năng chuyển đổi giữa chúng. Những thay đổi bạn thực hiện đối với ứng dụng trong bài thực hành này sẽ không ảnh hưởng đến hành vi hiển thị của ứng dụng đối với người dùng.

Bước 1: Thêm các phương thức callback của vòng đời vào TwoActivities

Trong nhiệm vụ này, bạn sẽ triển khai tất cả các phương thức callback của vòng đời Activity để in thông báo ra Logcat khi các phương thức này được gọi. Những thông báo

log này sẽ giúp bạn thấy khi nào vòng đời của Activity thay đổi trạng thái và cách các thay đổi đó ảnh hưởng đến ứng dụng khi nó chạy.

1.1 (Tùy chọn) Sao chép dự án TwoActivities

Đối với các nhiệm vụ trong bài thực hành này, bạn sẽ chỉnh sửa dự án TwoActivities mà bạn đã xây dựng trong bài thực hành trước. Nếu bạn muốn giữ nguyên dự án TwoActivities trước đó, hãy làm theo các bước trong Phụ lục: Tiện ích để sao chép dự án.

1.2 Triển khai các phương thức callback vào MainActivity

1. Mở dự án TwoActivities trong Android Studio, sau đó mở MainActivity trong Project > Android.
2. Trong phương thức onCreate(), thêm các câu lệnh log sau:

```
Log.d(LOG_TAG, "----------");

Log.d(LOG_TAG, "onCreate");
```

3. Ghi đè phương thức onStart() và thêm một câu lệnh log để ghi lại sự kiện này.

```
@Override

public void onStart(){

    super.onStart();

    Log.d(LOG_TAG, "onStart");

}
```

Để nhanh hơn, chọn Code > Override Methods trong Android Studio. Một hộp thoại sẽ xuất hiện với danh sách tất cả các phương thức có thể ghi đè trong lớp của bạn. Chọn một hoặc nhiều phương thức callback từ danh sách này sẽ tự động chèn mẫu hoàn chỉnh cho chúng, bao gồm cả lệnh gọi bắt buộc đến superclass.

4. Sử dụng phương thức onStart() làm mẫu để triển khai các phương thức callback vòng đời khác:

- onPause()
- onRestart()

- onResume()
- onStop()
- onDestroy()

Tất cả các phương thức callback có cùng chữ ký (chỉ khác tên). Nếu bạn sao chép và dán onStart() để tạo các phương thức callback khác, hãy nhớ cập nhật nội dung để gọi đúng phương thức trong superclass và ghi log đúng phương thức.

5. Chạy ứng dụng của bạn.
6. Nhấp vào tab Logcat ở dưới cùng của Android Studio để hiển thị bảng Logcat. Bạn sẽ thấy ba thông báo log hiển thị ba trạng thái vòng đời mà Activity đã chuyển qua khi bắt đầu.

```
2025-03-06 05:17:03.161 14981-14981 MainActivity          com.example.twoactivities          D  ----------------
2025-03-06 05:17:03.161 14981-14981 MainActivity          com.example.twoactivities          D  onCreate
2025-03-06 05:17:03.171 14981-14981 MainActivity          com.example.twoactivities          D  onStart
2025-03-06 05:17:03.172 14981-14981 MainActivity          com.example.twoactivities          D  onResume
```

2.3) Intent ngầm định

Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

1.1) Hình ảnh có thể chọn

1.2) Các điều khiển nhập liệu

1.3) Menu và bộ chọn

1.4) Điều hướng người dùng

1.5) RecycleView

Bài 2) Trải nghiệm người dùng thú vị

2.1) Hình vẽ, định kiểu và chủ đề

2.2) Thẻ và màu sắc

2.3) Bố cục thích ứng

Bài 3) Kiểm thử giao diện người dùng

3.1) Espresso cho việc kiểm tra UI

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

1.1) AsyncTask

1.2) AsyncTask và AsyncTaskLoader

1.3) Broadcast receivers

Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền

2.1) Thông báo

2.2) Trình quản lý cảnh báo

2.3) JobScheduler

CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

1.2) Cài đặt ứng dụng

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel